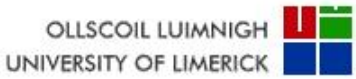
October 2007

Efficient Surveillance System

Janelle L. Tavares
Worcester Polytechnic Institute

Matthew Conway
Worcester Polytechnic Institute

Phong Cam Dam
Worcester Polytechnic Institute



Efficient Surveillance System

A *Major Qualifying Project* Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

Worcester, Massachusetts, USA

In partial fulfillment of the requirements of the

Degree of Bachelors of Science

on this day of

Friday, October 12, 2007

by

Matthew Conway

Phong Dam

Janelle Tavares

Advisor Prof. Donald R. Brown

Co-advisor Prof. Richard R. Vaz

Abstract

At the University of Limerick, we designed a proof-of-concept device for a low-power, low-cost wireless surveillance system, partially powered by solar energy. We designed a portion of the system - the camera node and the base station's user interface. The major functionality included capturing pictures based on motion detection and remote notification of this activity, which could also be viewed online. We designed this prototype to fill a gap in the market for low-cost, highly-scalable surveillance systems.

Acknowledgments

Our team is grateful to Prof. Richard F. Vaz and Prof. Donald R. Brown for giving us the opportunity of carrying out our Major Qualifying Project (MQP) at the University of Limerick (UL), Ireland. Special thanks to them for their guidance that led us to the successful completion of this project.

We would especially like to thank Seamus Clifford, John Harris, and Mark Southern at the Enterprise Research Centre (ERC) at UL for providing continuous support and assistance throughout our project.

We would also like to thank the staff of the Circuits and System Research Centre (CSRC) at UL for allowing us to have complete access to their equipment. Special thanks to Simon Effler for aiding us in understanding and resolving technical issues involved in this project.

Finally, special thanks to Charlotte Tuohy for her assistance and enthusiasm in ensuring we remained comfortable during our stay in Limerick, Ireland.

Contents

List of Figures

List of Tables

Executive Summary

One of the prevalent crimes in Ireland is theft of a motor vehicle. According to one statistic, Ireland ranks tenth among the all countries in car thefts per capita, falling just below the United States [14]. The prevalence of this crime is one of the inspirations for this project. The Enterprise Research Centre at the University of Limerick has noticed the trend of automobile crime taking place within the university's own campus grounds. Problems such as break-ins, theft, and vandalism have all been reported to have taken place in the campus's car parks, according to the Buildings Officer. These problems have led our project sponsors to consider the need for a surveillance system that could be installed in car parks, or other locations.

Closed Circuit Television (CCTV) security systems are commonly used to monitor car parks; however, the sponsor did not choose to consider the use of this type of system. Installation costs of a CCTV system would not only be high, but parts of the car parks would also have to be reconstructed to install the wires from each camera to the base station. One company gave a quote of €3,298 for a security system consisting of four colour fixed cameras, a 17" LCD monitor and one Digital Video Recorder - €1,147 for installation and €2,151 for components.

What our sponsors proposed was a low-cost, camera-based security system. Four students out of the seven, the mesh network team, worked on the wireless communication portion of this system. .We designed the remainder of the camera node and a portion of the base station. When designing the system, we considered the criteria the sponsors provided us: the camera system should sense human's movements, capture images, have its own power supply, load the images onto a website, and remotely notify a user of the new activity in the car park. After reviewing the criteria that we had to fulfill, we constructed a block diagram, Figure 1, to show to components in the system and their interfaces.

Figure 1 shows the basic functional areas necessary for this system. We researched components and modules to fulfill each block. At the conclusion of this research, we made decisions about what would suit this project. For motion sensing, we chose a passive infrared (PIR) sensor, as they are low power, that outputs a logic high voltage when it sensed a sudden change in temperature gradient - the motion of a person. In particular, the selected PIR sensor was AMN14111 from Panasonic can sense motion at a range of 10m. We chose the COMedia C328-7640 digital camera to fulfill the functions of the camera module. One of the important features was its JPEG compression engine; using this would reduce the time necessary to transmit the images and, therefore, the overall power consumption. We used a Transcend Security Digital (SD) card for the system's memory since it had a reasonable cost, a desirable capacity - 128MB, and a fast data transfer rate. The central processing unit that we used to control all of the above components was the MSP430F1611 ultra low power microcontroller. A 6Votls, 8.5Ah, sealed lead acid battery powered the system; however, we found that a battery alone could only power the system for about a week. To charge the battery, we connected it to a solar panel and charge controller from Silicon Solar Inc. The final portion of our surveillance system was the base station. We used a PC running Windows as the base station.

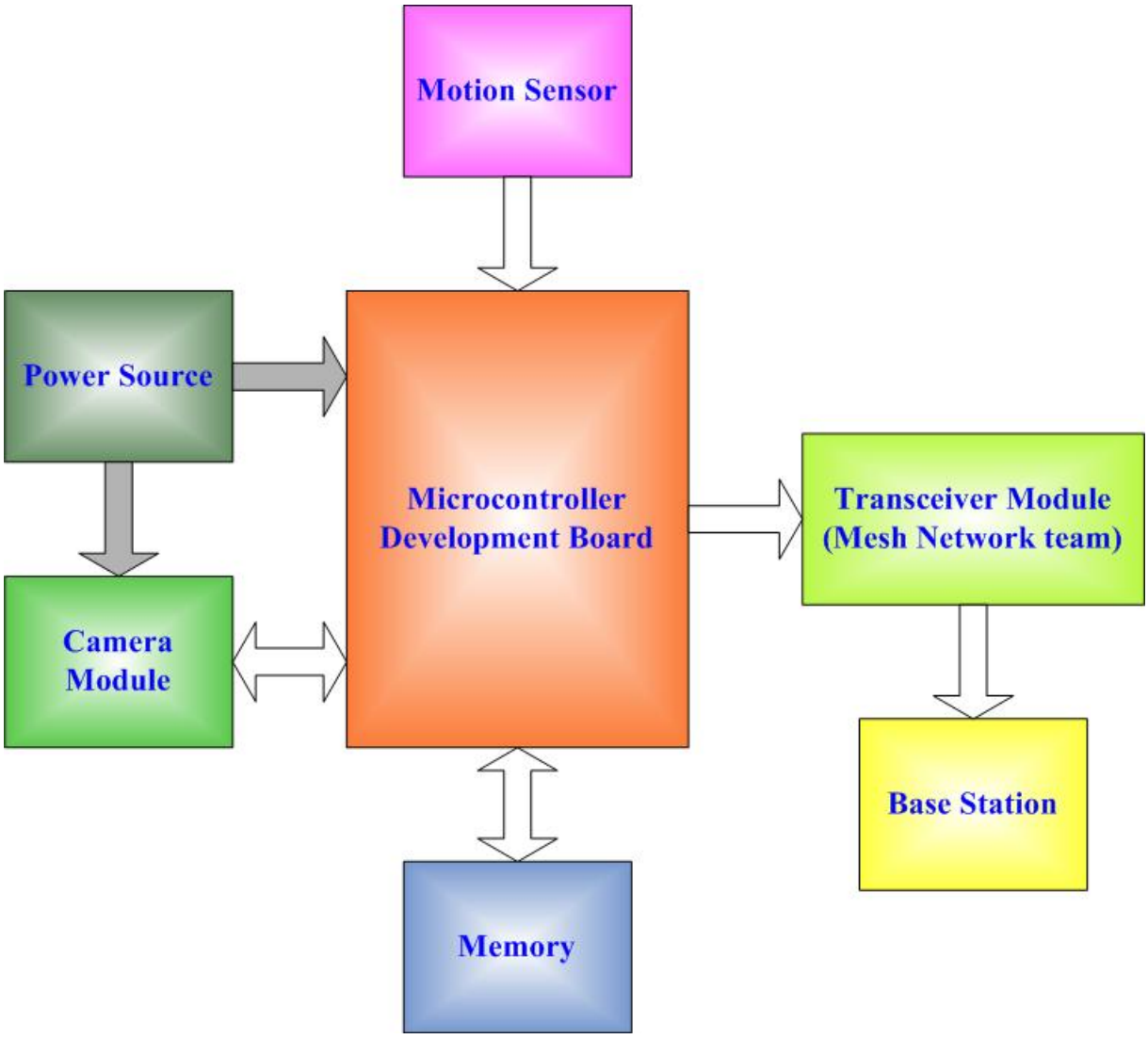


Figure 1: Block diagram of system.

After building and integrating the system, we tested its functionality. When the PIR sensor detected motion, it outputted a signal on an interrupt pin on the MSP430 microcontroller. This caused the microcontroller to wake from a low power sleep state and to send snapshot commands to the camera module. In order to be more power efficiency while transmitting wirelessly, as that consumed the most power, the camera module processed the image into a compressed format - JPEG. The microcontroller obtained this picture information from the camera module to store it in the SD card. Then, the microcontroller communicated with the transceiver in the mesh network team's design to transmit the pictures to the base station. The base station's microcontroller received the information and saved it in a specified file folder. With code written in PHP and C++, our portion of the base station transferred the JPEG files to a webserver, uploaded incoming images onto a website and generated alert messages in the form of emails with attachments and text messages. In addition, the user could add email addresses or mobile phone

addresses to the list of those notified with links on the website. Figure 2 shows a screenshot of this website, with those links at the top.

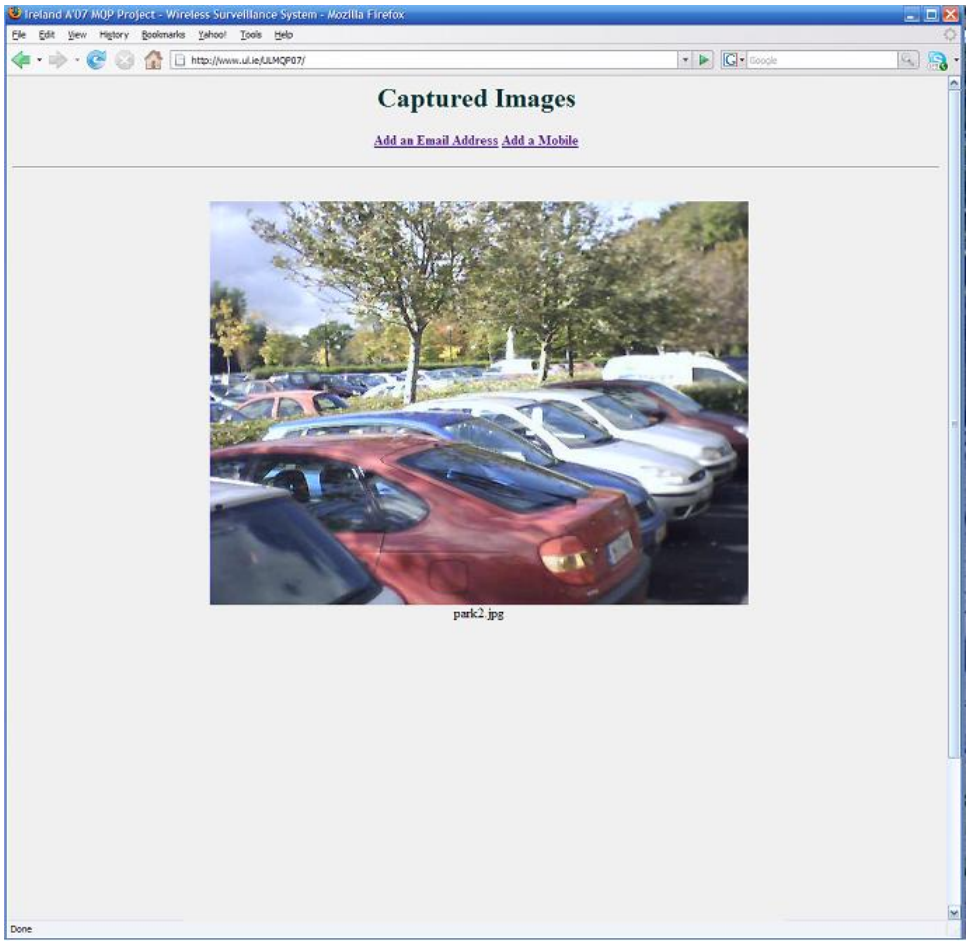


Figure 2: Screenshot of Base Station Website.

While we were not able to document the behavior of the power supply over the long term, we calculated that the power supply could last for about a week if the camera node was active half the time, and in sleep mode the other half of the eight hours we expected it to be monitoring an area. With the number of charge cycles, it could last for three or more years; however, the battery would be completely drained in three of the winter months as the sun in Limerick is not intense enough. As the power supply would be self-sustaining the other nine months, the user would have to charge the battery once a month for those three months to sustain the power supply for several years. A more powerful solar panel or additional means of energy harvesting in the future would help to increase the lifetime of the battery. Figure 3 depicts the power supply and camera node in their final prototype containers.

Along with the power supply lifetime, we fell short with one other requirement. The sponsors desired that the camera take two pictures per second for five seconds after the motion trigger. We achieved two pictures in 34 seconds after the trigger at a resolution of 640x480. While we did not foresee a solution for meeting that requirement, we did find that using the maximum

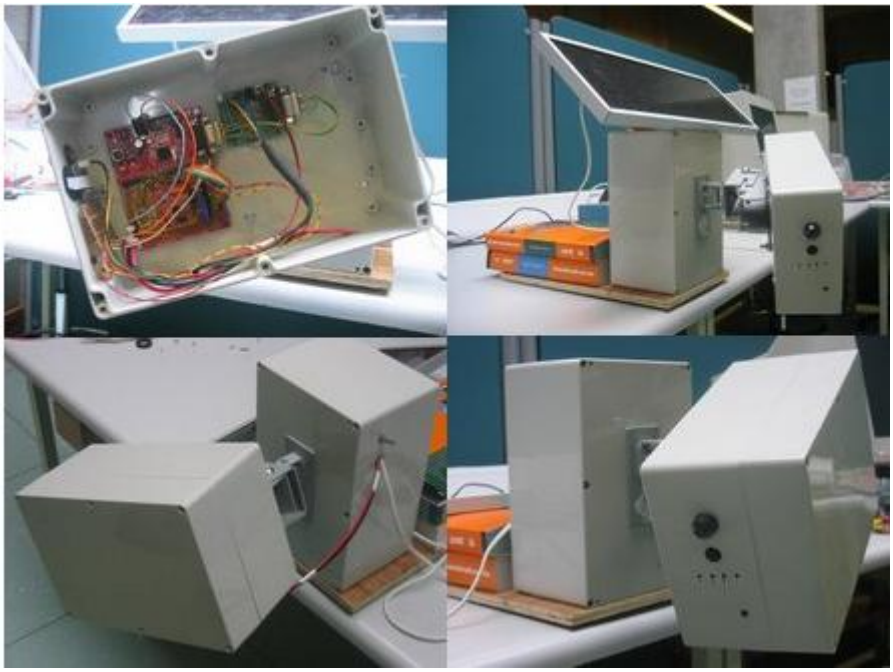


Figure 3: Pictures of the Final Prototype.

compatible baud rate interface, 115,200 bps, between the camera and the MSP430 could greatly increase the speed of the system. Another alternative is to consider a different camera module which supports SPI interface because the fast data transmission in SPI mode could satisfy the requirement.

In terms of the cost effectiveness of this system, we considered the cost of the system as if we were producing in bulk. The cost of raw materials for a camera node and base station per 1000 quantity were €171.53 and €88.98, respectively. Installation would consist of mounting each node onto light posts already located in car parks, and this would factor into the overall cost as labor. Though this estimate does not include potential software costs, we would not anticipate that they would be substantial. These figures are considerably smaller than the quote we were given. For four cameras, our system would cost about €775, without installation. For a figure closer to the quote we received, our system with twenty camera nodes would cost about €3,500. Refer to Appendix B for these calculations. In addition, the customer would save more money over the long term as our surveillance system would not run off of AC power, but rather harvest the energy of the sun. At the conclusion of the project, we found that it was feasible to implement a low-cost, low-power surveillance system and that we were mostly successful in designing that system.

1 Introduction

One of the most common crimes in Ireland is theft from or of a motor vehicle. Theft from a parked motor vehicle is the third most frequently reported crime, after burglary and theft from a store, according to the Garda Annual Report [40]. Between 1995 and 2000, the incidences of motor vehicle theft increased 29% in Ireland and 37% in Northern Ireland, as compared to a decrease of 34% in England and Wales and a decrease of 31% in Scotland [13]. While there was a decrease of 21% between 2000 and 2005 in Ireland, incidences of these crimes are on the rise again [39]. Reports of theft from a parked motor vehicle increased by 3.7% in 2006 compared to 2005, while the report of all crime only increased by 1.4% [40].

The University of Limerick experiences problems with crime in its car parks. Robert Reidy, the Buildings Officer with the Buildings & Estates Department, stated that the types of crimes reported in car parks include car break-ins, car theft, and car vandalism. Other concerns are assault, liability with accidents, unauthorized use of the car park, and evasion of car park charges. The project sponsors have noticed an increase in car park crimes with new construction projects on campus. Car parks on the perimeter of campus provide an easy access point for persons from off campus; thieves or vandals can quickly retreat off campus to an area in which campus security no longer has jurisdiction.

In particular, the Foundation Car Park was an inspiration for this project's sponsors. They sought to find a security system suitable for the car park. Typical systems can provide security by monitoring who comes and goes (access control) and monitoring what occurs in an area (surveillance). Access control relates to security barriers and a vehicle registration database. The aim for this security system was mainly surveillance. It alone can be particularly effective at deterring crime; installing a closed circuit television (CCTV) security system has been shown to reduce incidence of vehicular theft and vandalism up to 34 percent [36]. Available products for surveillance of a car park include fixed cameras; pan, tilt and zoom (PTZ) cameras; specialized cameras to take pictures at night or capture license plate numbers; video recording; and remote monitoring. Typically, these systems require AC power. The Foundation Car Park does not have power junction boxes currently, so this would be a consideration before a system consisting of the above components could be installed. The cost for running power lines and installing power boxes is prohibitively high, such that the project sponsors rejected this type of power. A Northern Ireland security company, Secure Vision Systems, Ltd., quoted the group a figure of €3,298 for a security system with only one 4-channel Digial Video Recorder (DVR), four colour, fixed cameras and one 17" LCD monitor; this costs includes installation, at €1,147. In addition, some wireless surveillance system are available. They are generally intended for residential use, as they have limited range. These are not suitable for an application such as a car park.

With those costs in mind, the sponsors of this project presented to our group of students the challenge of designing a wireless surveillance system that would be cheaper than a system requiring power junction box installation. Not only would this system be intended for a car park, but other areas in which customers may want surveillance. Wireless has two implications

here; the system must rely on power from only DC sources and the surveillance system must be able to communicate wirelessly so that information gathered can be monitored by the necessary individuals.

Specific design issues with this project revolve around cost and power efficiency. Costs include the actual cost of prototyping (materials), estimated installation costs (labor), and estimated maintenance costs (labor to retrieve batteries, install new batteries). Designing a power efficient system reduces the maintenance costs over the long term, and therefore, the overall costs. Using less power requires that electronic components are drawing current only when necessary. This type of system could then utilize a motion sensor to control when components are powered on. Also, the components must use as little power as possible while powered on. For instance, taking a series of still images uses less power than capturing video. This information must also be made available to security personnel. Therefore, it must be efficient to transmit data, so packaging the data is another concern. A means of harvesting energy from the environment would also extend the life of the power supply.

Reviewing the current market for surveillance security systems, power efficiency decreases significantly with the increase in area surveyed; systems intended for a larger area require utility power, while systems that transmit wirelessly and that do not require utility power only cover a small area. Therefore, the gap in the market is for a low-power, low-cost system which can gather information about a relatively large area. Reducing installation costs significantly by not requiring any physical lines, our system is a major improvement over CCTV. In addition, this system would be able to cover an area of variable size, potentially much larger than those areas covered by existing home security systems. In this way, improving upon both types of systems would create a new niche in the security system market for a low-power, low-cost system that is still powerful in terms of range.

With this project, we sought to prove that this type of system is feasible by creating a prototype. The system we created consisted of a camera, a PIR motion sensor, microcontroller, memory, software for the base station and power supply with a recharging method. Another group of students from WPI design the wireless communications portion of this system; we will refer to them as the mesh network team throughout the report. Our system could take pictures based on motion trigger and prepare those images for transmission. With the charging method, the power supply could theoretically power the system for more than three years, though a user would have to charge the battery once a month during three of the winter months as the sun is not intense enough in Limerick. Also, we provided a website run from the base station of the wireless system for viewing images. The base station recognized new images and notified any number of persons of new activity captured by the cameras. In terms of the cost effectiveness of the system, we projected that the production version of our system would be about five times cheaper than the system we were quoted. In this report, we describe our engineering process and our surveillance system design.

2 System Requirements

In this chapter, we describe the requirements for this proof-of-concept prototype of a security camera surveillance system. It also served as a guideline for the team to carry out the design process and provides the audience with an understanding of the direction of the project.

The University of Limerick proposed the project to two groups of students. The following is an overview of the main project as described by the sponsor:

> The project will require the design and development of a wide area nodal/cellular wireless security camera system that is operable with very high energy efficiency, i.e. with very low power consumption. Each individual camera will be woken up by its PIR movement detector (mounted below the camera) and will then take a few pictures at a frequency of about two pictures per second for, say, 5 seconds. The output of the cameras will be sent to the next camera (<100 meters away) which will pass the image onto the next camera until it reaches the base station which may be some distance away. The base station will send the pictures to a website and to a mobile phone.

Our group sought to design the portion of the project with the camera node and the user interface of the base station. We undertook the portion of the project that the sponsor further explained with this description:

> The design and development of the base station, the external interface and the appropriate software for the base station, website and the mobile phone. The website and mobile software only need to display the images in order to prove the concept. Activities include: development of an energy efficient camera system including PIR [passive infrared] sensor, camera, power supply, solar cells, physical/logical interface to mesh network subsystem and associated documentation.

Based on the main project's description and the sponsor's suggestions, we were able to derive the following system requirements:

- The production cost of each camera node should not exceed €250.
- This security system must be highly energy efficient, i.e. the battery supply should last a least for a year.
- The camera nodes should be:
  - within about 100m of each other, the maximum transmission range set by the FCC
  - capable of detecting human's movements
  - capable of taking photos upon detecting that movement
  - capable of transmitting and receiving photos

- The base station should be capable of:
  - receiving and storing images from the cameras
  - uploading photos to a server for a website
  - sending notifications, which are photos and/or messages, of all activity, preferably to a mobile phone

After a few discussions with the sponsor, we made the following decisions for the initial product specifications of our particular project:

- Camera node
  - *Ability to detect movements:* A motion sensor would be used for detecting movements.
  - *Ability to take photos:* A digital camera should be able to take still photos as soon as activity triggers the motion sensor. Upon that trigger, the camera should take two pictures per second for five seconds.
  - *Storage for taken photos:* The storage should be a memory device that can hold the photos temporarily or permanently.
  - *Battery powered and energy efficiency:* As the system would not have access to dedicated AC power, the camera node should operate efficiently on battery power to reduce maintenance; the battery should last for at least a year.
  - *Central controlling unit:* This unit would control the entire operation of the camera node.
- Base station
  - *Storage for receiving photos:* This storage should be a computer hard drive and its capacity should be large enough to store the photos received from the camera nodes.
  - *Ability to upload photos to website:* We should design a website for uploading the photos from the hard drive.
  - *Ability to send messages or photos to mobile phones:* Notifications should be sent to users' mobile phones of activity in the car park.

2.1 Prior Art Surveillance Systems

After defining the requirements of the prototype and the goals for the project, we began researching prior art to gain an understanding of how this prototype can be designed and implemented. This section describes information the group found during the research stage and what impact it had on how we decided to proceed with the project.

2.2 Prior Art Surveillance Systems

The following section describes prior art in regards to existing surveillance systems. We present several systems in order to determine the market with which our surveillance system would compete. In comparing some of the technical information about these systems, we were able to develop more details for the specifications of the new system.

The first camera system is the TransTech wireless four camera security DVR system. The cameras are equipped with night vision, allowing for clear pictures at distances of up to 13.7m. Additionally, this system transmits color video until it detects low light conditions, in which case it uses infrared night vision to illuminate the space. The range of wireless transmission to the base station is the longest range that the Federal Communications Commission (FCC) allows, making it possible to spread out the cameras and cover more ground. Through the use of the software included, one would have the control to set the level of sensitivity for motion detection or to set specific times to record, customizing one's level of security. A computer would store the videos with time and date stamp to help keep track of files. The system can also output JPEG still images, as well as video. Once the videos are stored, one could view them over the Internet with a PC or with a cell phone that has browser capabilities. With this capability, one could look over his property from just about anywhere. Four cameras come with the surveillance packages, all of which run on AC power. This system is rather affordable at €232.08.

This system has many features that make it desirable to use in different applications. There are, however, drawbacks to this system due to features that it does not have. One of these drawbacks is that the cameras cannot run off battery power. Therefore, they cannot be placed in areas without utility connections. Another is that the system cannot be expanded through adding more cameras; the largest number of cameras that the system can handle is four. Finally, the cameras all transmit images back to the receiver; they are not able to transmit to each other, decreasing the overall area potentially covered by the system [44].

The next surveillance system that we reviewed was the wireless 4 camera IR weatherproof system with a CPCAM 250GB DVR and a CD-RW from 123securityproducts.com. Without obstructions, the included cameras are able to transmit up to 91.4m during light conditions. With the system's infrared LEDs, the cameras are able to take pictures in the dark with clarity. This again allows better opportunity to survey areas in all conditions of light. The network DVR uses MPEG-4 compression in order to efficiently transmit the data. The camera network can use 4 channels which allow the cameras to operate within the same area or same system. With this system, one could closely watch over one area or spread out the cameras to cover more ground. The cameras run on utility power which does restrict the locations of the transmitters. The package comes with four color cameras, four receivers, and one digital video recorder with 250GB hard drive and CD-RW.

As with the previous system, this one is not expandable; the maximum number of cameras is set at four. There is also no transmission of data between cameras; the data is sent back to the receivers at the DVR. Additionally, this package does not have the capability of taking still

images, it only records video. Finally, this system is not able to remotely notify a user of new activity and is quite expensive at €903.72 [51].

The Mini-AirWatch 4 is another affordable security camera system at €196.91. The transmitters can broadcast up to 30.5m indoors or 91.4m outdoors unobstructed. The video that is received from the camera transmitters is fed into a TV or a VCR to monitor activity. The cameras are small and can be placed anywhere since they are equipped with batteries. This allows one to be able to monitor even remote areas for a period of time without worrying about AC power availability. The package comes with four cameras that are able to record video and one video receiver. There are four channels available with this surveillance system and each camera can be controlled with a remote. Sound is also recorded, in addition to the color video, further enhancing the security level.

This security system has many fewer features than the two previous ones described. Like the others, though, this one is not able to be expanded with more cameras and there is no transmission among the cameras. The system is designed to be used for monitoring purposes, not capturing still images. The receiver is not able to be connected to a PC and cannot notify a user in any way of new activity. Finally, while the cameras can run on battery power, the drawback is that the life of the batteries is limited to about 2 to 3 hours [38].

The last security camera package that we reviewed is the Panasonic network camera package. The system uses thermal motion detection in order to detect intruders. The system operates on IEEE802.11 wireless and the cameras have a built in web server/IP address with a Viewnetcam.com address. Once images have been recorded, the data is stored and recorded to an SD memory card for later viewing.

The receiver for this system has a TV interface for the network and can connect to the cameras and router with an Ethernet cable or with a wireless router. The system can link to the cameras over the Internet to monitor activity while away. The built in thermal sensors trigger the system to email up to three addresses. The email can also be set to have JPEG images attached or no images attached. To mechanically control the cameras, there is a remote control to change the pan and tilt angles. With all of the features included with this system, however, it does not have an option of running on a DC battery source. On the other hand, although only four wireless cameras can operate at once, there is room to expand with four wired cameras.

More than any of the other packages we reviewed, this one costs €1,044.36, although it includes many features. While it is possible to expand the system with wired cameras, no more than four wireless cameras can operate on the system at once. Like all the systems mentioned, the cameras are not able to transmit to each other, only back to the main receiver. Also, the cameras cannot run on batteries; they are designed to run on AC power [33]. Table 1 summarizes these reviewed systems.

While the systems we reviewed reflect a sample of the market, the set of design requirements for our surveillance system was still unique. While still being wireless, our system required that it run on battery power for considerable lengths of time and that the system remotely notify

Table 1: Summary of Prior Art Surveillance Systems.

System	**Transtech**	**123SecurityProducts**	**Mini-Airwatch 4**	**Panasonic**
Range/Viewing Capabilities	• infrared night vision • image capture up to 13.7m • transmission range longest allowed by FCC	• transmit data up to 91.44m • infrared LED's for dark conditions	• transmit up to 30.48m indoors • transmit up to 91.44m outdoors	
Data Format	• video • JPEG	• MPEG-4	• video	• video • JPEG
Method for Retrieving Data	• stored on main computer • Internet • cell phone	• DVR	• TV/VCR	• TV • Internet • email
Power Method	• AC power	• AC power	• AC power • battery option	• AC power
Capacity of Network	• 4 cameras	• 4 cameras • 4 receivers	• 4 cameras	• 4 wired cameras • 4 wireless cameras
Other Features	• variable level of motion detection • time/date stamping	• operates on 4 channels • sound recorded	• control cameras with remote • web server/IP address	• IEEE802.11 • remote control • email notification
Cons of System	• only AC power • not expandable • no data transfer from node to node	• only AC power • not expandable • no data transfer from node to node • expensive • only records video • no remote notification	• not expandable • no data transfer from node to node • expensive • only records video • no remote notification • battery lasts only 2-3 hours	• not expandable • no data transfer from node to node • expensive • no battery option
Price	€232.99	€907.29	€197.69	€1,048.49

users of activity - a combination that we did not find in the market. We concluded that a system to meet our design requirements does not exist, and therefore our design project filled this gap in the wireless surveillance security system market.

2.3 Prior Student Research

In researching existing systems on the security market, we were able to gather information about what features were standard and desirable in the wireless security system market. This section describes what we discovered by researching projects other groups of students have previously completed. We hoped to gain an understanding of how the desired behavior of a surveillance system can be implemented. Also, this research provided some perspective on the expectations of performance for current technology, as well as design considerations.

One research team of interest was from the Electrical Engineering and Computer Science (EECS) department at the Massachusetts Institute of Technology (MIT). An article published in 1998 in the IEEE Xplore explains their approach for designing a low-power wireless camera

[12]. The article described how the student team sought to reduce power consumption of a camera network, which consisted of base stations and wireless video camera nodes with embedded power supplies. Their approach to minimizing power consumption involved varying the quality of service and energy consumption, depending on user preferences; their approach skewed the load of processing video information, as well as minimized leakage current, all to optimize power consumption. Varying quality of service and energy consumption relates to a compromise of one for the other, the degree of which depends on user preferences and the conditions. Part of quality of service is the amount of video compression; an example of varying the quality of service would be reducing the amount of video compression while the wireless camera node processor was in high demand at that point. In this way, the image quality would be higher, but it would require more time and resources from the wireless communication portion of the network. Also, depending on the conditions, an adaptive power supply helped the circuit meet operational timing constraints by using power supply feedback.

Processing video involved running several algorithms. The most power-hungry was motion estimation between frames. The student team proposed performing this calculation at the base station, as it presumably would be powered by an AC source, while the node performed the other calculations. In this way, the wireless nodes would send several video frames to the base stations. Then, the base station would run algorithms to find the motion estimation between the received frames. When the base station returned the information, the wireless camera node could then finish the processing, continue recording video and repeat this cycle.

Last, the student team at MIT considered how to further minimize the power consumption of the wireless camera nodes during the sleep mode. As they estimated the nodes would be in sleep mode about 90 percent of the time, minimizing the power consumption during this state would have a major impact on the overall power consumption. First, the nodes should enter this state whenever there is a low amount of motion. In this state, the student team sought to aggressively cut supply voltage as much as possible for efficiency. Also, they proposed using Multiple Threshold CMOS (MTCMOS) technology, which incorporates a high threshold transistor, to reduce leakage current. From this article, we can appreciate the importance of an efficient sleep state, the importance of utilizing technology to be more energy efficient, and the impact of certain compromises to improve the overall design.

Another article of interest came from an employee of Avago Technologies in San Jose, CA, and a professor in the Department of Electrical Engineering at Stanford University. The sensor network described in this article consisted of camera modules, image processing and wireless communications [3]. The network used an algorithm that compared frames of data from the camera to determine an event to record. Their article mainly focused on Application Program Interfaces (API) for this wireless image sensor network. These APIs were a library of MATLAB-based programs implemented to simplify the user interface. The APIs consisted of the basic functions (open, initialize, capture frame, receive, send, close) for multiple types of interfaces. With them, a user would not have to configure the interface between the image sensor

and its controls; a user would not experience the difference between, for example, a Universal Asynchronous Receiver-Transmitter (UART) interface or a Serial Peripheral Interface (SPI). While this is a very specific concern, it is important for the group to take away from this article the value of a smooth and easy-to-use interface for the customer.

The final article we reviewed came from the Institute of Computer Science at the Freie Universität in Berlin, Germany. Published in 2004, the research of this group of students pertained to sensor networks (temperature, light, sound, vibration, IR, and image) that could be applied in several different scenarios, such as office building environmental control, wild-life habitat monitoring and forest fire detection [21]. The network consisted of sensor nodes and embedded web servers (EWS), which acted as base stations. They assumed in their design that the EWSs would have an AC power source. They controlled the nodes with MSP430 processors, while they used ARM processors for the EWSs.

The boards for the nodes were quite customized in that they had several sensors (listed previously) and they designed the boards specifically for low power. For example, during sleep mode, all sensors would be off. An event would trigger an interrupt to power on parts of the circuitry; there was no polling. Though the image sensor module from COMedia required one second at 52mA to start up, the research team made the calculated trade off to keep it off while in sleep mode. Notably, the major energy source for these nodes was not a battery, but scavenging energy from the environment using vibrations, temperature gradient, and solar energy. Instead of using only a battery, these methods would replenish the charge in a 1F capacitor. With this set up, the capacitor could power about 420 cycles of sensing and sending data. With all sensors running, the board only drew 12mA of current. While transmitting data - such as JPEG compressed images - using radio frequencies, the draw was less than 8mA on average. In sleep mode, the circuits only drew 8μA to run the clock. Using those numbers and a AA battery as back up power, it would last theoretically five years with a duty cycle of one percent.

Not only was this system's power efficiency impressive, but also its ability to be customized. The research team had set up several channels to configure the system. First, they projected that the system would be set up using existing Internet and telephone infrastructure, so that this system could access the Internet and use mobile phone communications. With a personal computer (PC) or a personal digital assistant (PDA), one would be able to connect to the system to configure it. Also, it could be configured through the mobile phone network with SMS text messages. Other software capabilities included analysis of sensor data, control of peripherals, notification by SMS text message of sensed events, and the ability to poll the sensors about the current environment. This paper described a complex system with many features and customized circuitry. While we need only prove the concept for our project, we can use their many features to consider a suitable implementation for our surveillance system.

3 Design Approach

To provide the audience with an initial understanding of the overall design, this chapter presents the possible system-level approaches that we could have used to design this prototype. This chapter explains the system in terms of blocks, first for the camera node and then for the base station.

3.1 Camera Node

The purpose of this section is to describe two major design approaches for the camera node. Each design option includes description of the major sub-systems, the advantages and the disadvantages, and the evaluation regarding the system's efficiency, as well as the design's feasibility.

3.1.1 Individual Components Approach

With our fundamental understanding of the camera node's functionality, we determined the first system-level design approach. Before describing the design option, it is necessary to review how the camera node operates. Generally, the camera node must be able to detect movements, take snapshots, and store the images. Initially, we considered achieving these functions with individual components. Specifically, a motion sensor could be used to detect suspicious activities; a camera could be responsible for taking images; a memory device could be used to store the photos; and most importantly, a microcontroller could be the central unit that controls and manages the operation of the whole system. In addition, the camera component consists of its own components: an image sensor, an image processor, and a compression engine. The image sensor could interpret the visual images to electrical signals that the image processor could then process. The compression engine could convert the raw image data produced by the image processor to a desired image format. The output of the compression engine could be connected to the microcontroller for further processing.

In this design approach, we would have to build the camera node by assembling the individual components together. Figure 4 illustrates how all of the major components of the camera node would be integrated into a complete system.

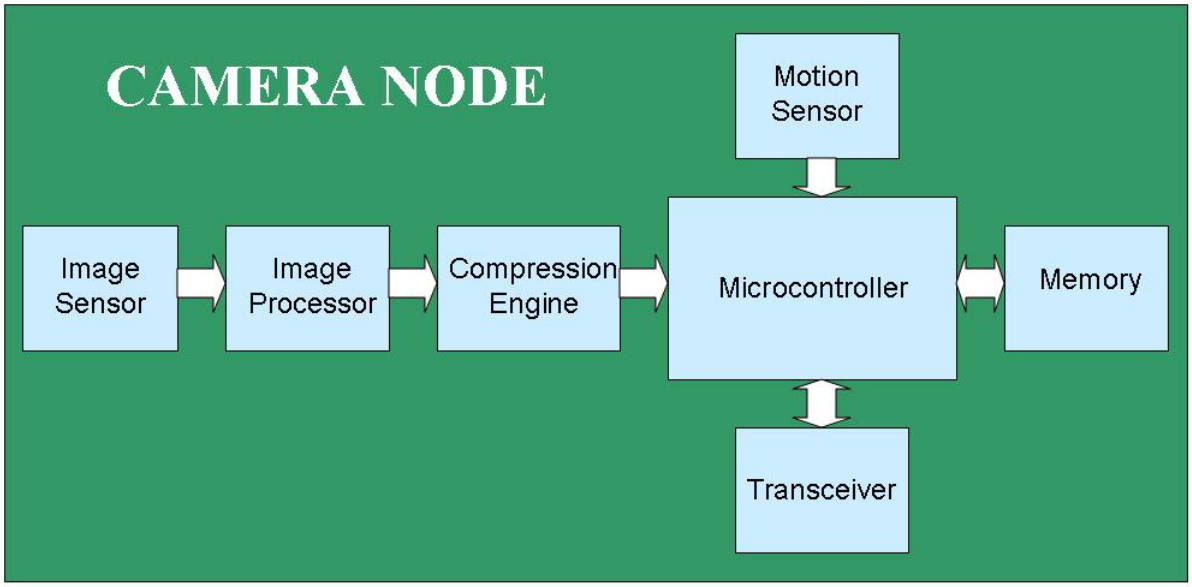


Figure 4: Design approach with individual components.

One advantage of this design option is that we would gain a good understanding of each individual component, and thus be able to customize each component to our specific needs. Moreover, we could have been be able to obtain free samples from the manufacturers. Another possible benefit is that we could be able to achieve greater power efficiency by selecting only components with the lowest power consumption.

Although this design option would satisfy the camera node requirements, there are many drawbacks that seem to overwhelm the possible advantages and make the design unfeasible within the time frame allotted for this project—ten weeks. The major issue is that we were initially unable to define the interfaces among the illustrated components. This design approach would require us to determine a suitable interface between all components and to configure each. Another problem is that it is difficult to find an image processor and a compatible compression engine. Furthermore, this design option might involve a great amount of wiring and difficult soldering; the components may have many pins or be small and difficult to handle. Generally speaking, we would encounter numerous obstacles if we tried to implement the system using this design approach due to limited available resources and our time constraint. With all these tasks considered, we found this approach to not be feasible.

3.1.2 Modules Approach

We considered a second design approach after analyzing the market for individual components. By using pre-fabricated modules, we would intend to minimize the complexity of the design process and maximize the likelihood of success. Figure 5 illustrates the design approach.

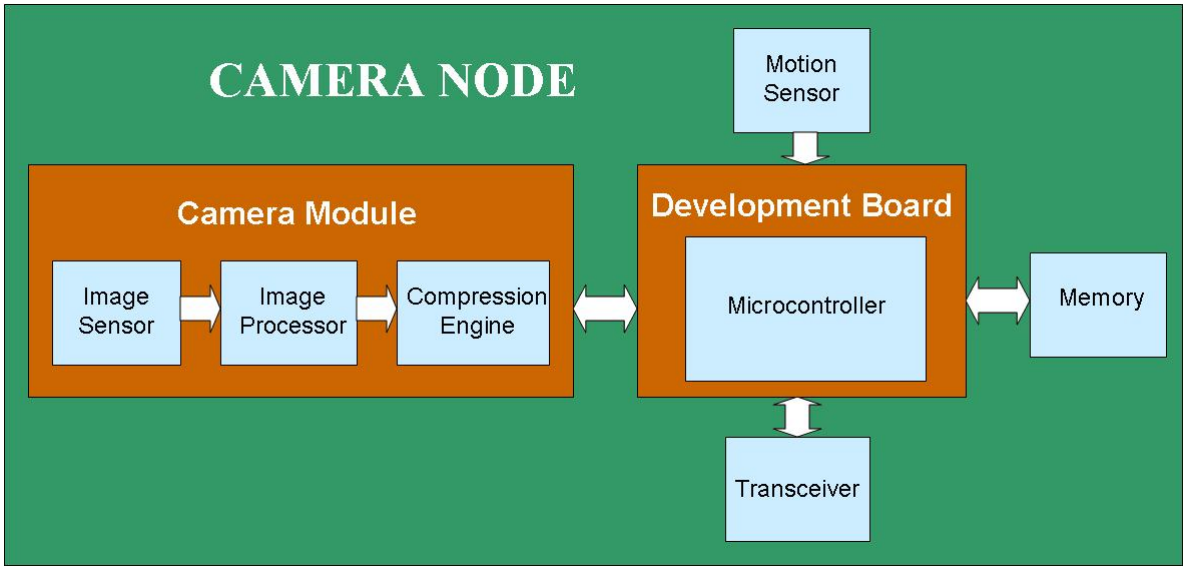


Figure 5: Design approach with pre-fabricated modules.

Upon checking the market for modules for this project, we found that camera modules already exist with an image sensor, an image processor, and a compression engine. Furthermore, the modules usually have a common type of interface, which would simplify the interfacing issue. We found that development boards, which incorporate a main microcontroller and many additional components, are readily available with pre-built external interfaces for peripherals as they are commonly used for teaching and researching. Moreover, as they are quite common, the cost is relatively low, such that building a board from numerous discrete components becomes unnecessary and unreasonable. Figure 5 depicts the system as it would be implemented with a camera module and a development board.

We must still define the interfaces remaining with this design approach. The interface between the motion sensor and the board is relatively simple. The challenge was to identify the following interfaces: the interface between the camera module and the board, the interface between the transceiver and the board, and the interface between the memory and the board. At this stage, we were certain that these interfaces were much less complicated than the interfaces in the first design option. A potential drawback to this approach was that the system might not be as power efficient; the modules may have unnecessary components that draw current. If it was possible to disable the unwanted features either by using software or by modifying the hardware, this power consumption would not be a major concern. Unlike the individual components design approach, complicated wiring and soldering could be avoided. Using this design approach would afford us more time to spend on writing software code, testing, and debugging. In general, the drawbacks were insignificant; the potential time saved and increased ease of implementation were more noticeable.

3.1.3 Determining Camera Node Design Approach

After defining our two design approaches, we chose one and proceeded with designing the prototype. The most important consideration when choosing a design approach was whether or not we would be successful in the allotted amount of time. To summarize our evaluation of the design approaches, we constructed Table 2. It reviews the pros and cons of each approach.

Table 2: Comparison between two camera node design approaches.

	Individual Components	**Modules**
Pros	• Exact desired functionality • Less expensive • Manageable power consumption	• Internal interfaces already configured • Existing external interfaces • No complicated wiring or soldering • More time for coding, testing and debugging
Cons	• Complicated internal interfaces • Complex wiring and soldering • More time consuming to assemble	• Limited choices for modules • Less power efficient • Relatively expensive

In this project, the design's complexity was the primary concern as an increase in complexity may have decreased our likelihood of success. Table 2 indicates that each approach had both advantages and disadvantages. At a glance, the list of pros for the modules design approach is the longest. The notable cons for the individual components approach included plenty of undefined and complicated internal interfaces. Assembling each interface would be especially time consuming. Although the cost would be minimized with this approach, this factor did not outweigh the benefits of using the modules design approach. After carefully considering the feasibility of the design options and our available resources, we concluded that utilizing a pre-fabricated camera module and a development board was the more suitable option for this project.

3.2 Base Station

The base station would perform several functions for the entire system. One of its functions would be to store images sent from camera nodes. Also, it should contain the software for the user interface. Through the base station, the user should be able to view the activity that the cameras have captured in the surveyed areas. The better the interface for viewing these images, the better security personnel may be able to respond to a situation. To set up a means for users to view images, we considered several aspects that would be involved in that process: storing image files, web hosting, website design, and alert messaging. As file storage is the only hardware aspect of this portion of the project, it would be difficult to describe the design approach wholly

in terms of individual components versus modules, as previously. Therefore, we organized the descriptions of our design options based on a different criterion—whether or not the aspect of the design was customized by the group or by an external service.

3.2.1 Customized Approach

The first general path for designing the base station was to do as much as possible without aid from outside of the Enterprise Research Centre. In this way, the design would be as customized as possible, but must be completely created ourselves. In this section, we describe the options for creating custom hardware and software for the base station.

The first concern for the base station was hardware. The base station is basically a computer that acts as the hub of the security network. The main function for our project was storing received image files; this hardware also had to run the website and its software. We also took into consideration customizing the other hardware components (processor, RAM) of the base station, as they would affect overall performance. The network would demand more performance from the base station as traffic increased. The speed at which the base station could access memory and retrieve stored images would have to be able to accommodate more camera nodes on the network and more users accessing the website. In customizing these components, we would seek to maximize that performance proportional to that demand. For example, in choosing a hard drive, we would specifically be interested in the capacity of the drive and the speed of its interface.

The base station would also have to run a website, not just support accessing image files. Running a website involves web hosting and website design. Web hosting includes creating an address for the site (often a registered domain name), storing some files for the web site, and managing the demand for resources from Internet traffic. The Enterprise Research Centre did not have the resources for the group to do this by itself, however, the University of Limerick did. Its Information Technology (IT) department supported groups within the University with web hosting by providing an Internet address (a portion of its own domain name) and its resources. We found this option to be the only relatively local option for web hosting.

The website is the portion of the base station that would be the actual user interface. It should allow the appropriate security personnel to access images of the car park from a computer on or off campus. For convenience, the website should be dynamic, in that it would update itself as the base station received new image files. We could accomplish website design without outside resources. One method would involve creating the website with a scripting language. As we did not have experience with scripting languages going into the project, we would have to use available resources, such as tutorials and reference materials. In addition, content management systems were available to help beginners; these systems are programs with libraries of tools meant to help a web developer manage designing a website. While these programs have been written externally to the group, we would still have to generate all code for the website.

The base station would have to support notification of security personnel about new activity in the car park, wherever he or she might be. We found several means available to send messages using the Internet and existing mobile phone features. In the least, emails could be sent to security personnel with attached images. One could achieve this function with short scripts. These messages would have to be viewed at a computer workstation, however, unless the security personnel had Smartphones (a phone that behaves much like a PC) with email forwarding. Another option would be to send a Short Message Service (SMS) format text message - which does not support multimedia - to a mobile device; one could send this type of message much the same way as an email, as service providers set up email addresses specifically to communicate between computers and mobile phones. Sending messages with multimedia could not be accomplished without external aid. We described that option, and the other options for setting up the base station with external aid, in the next section.

3.2.2 External Service Approach

In setting up the base station, we considered the benefits of using external services. In the end, these services could save us time and effort, and potentially generate better results. In the same order as the previous section, we described the external service options for fulfilling each base station requirement in this section.

If we were not customize the base station hardware, we would use an existing personal computer, as it was. In this way, we would use its installed operating system and add any necessary software to support the website. The operating system would be the interface with the portion of the base station that would receive image files. Only the installed hard drive would act as the available storage for those images. This set up would take little time or effort to accomplish.

We could procure services from professional companies for web hosting. For a monthly or annual fee, these companies would provide a unique domain name, and often other services such as Email addresses and web space storage. The fee would increase with the number of resources desired. One drawback to this option would be a lead time before these services could be used after the initial transaction.

After we established the website address, we could then procure the services of a website development company. As it would be professional service, the company should be able to provide all necessary dynamic content and support to fit our requirements. Depending on the complexity of the services provided, we could have to pay an initial fee and several monthly fees while they designed the website. The lead time before the website would be available could be much more substantial than the lead time for a web hosting service. Another drawback to this option would be that any tweaking of the site would require additional time to contact the company and request changes. Therefore, the site would be relatively static after set up. In the future, a development group may want to change the website for a production prototype, and website development choice could potentially be difficult to update.

Concerning alerts, we would be able to enhance our capabilities of sending alert messages to security personnel by using external services. In purchasing the service of a Multimedia Messaging Service (MMS) gateway, we would be able to send messages with multimedia to mobile device. Using this service, security personnel could have more complete information about new activity in the car park wherever he or she would be; with the images, security could react appropriately to whether or not the images depicted a crime in progress. Unfortunately, only certain mobile phones can receive messages with multimedia, though it is becoming an increasingly common feature.

When choosing a design approach for the base station, we foremost needed the design to meet the functionality requirements, and then we needed the approach to be an efficient use of our available resources—time and money. We found that neither customizing all options nor using external services was a blanket solution for designing the base station; a combination of the two would best suit this proof-of-concept project. By choosing the custom options for web hosting, website design and sending text alerts and using the existing computer for base station hardware, we would be able to stay within budget and use our time effectively to design the base station. After making decisions about the general design for the camera node and base station, we made a schedule for the remainder of the tasks for this project.

4 Project Planning

This chapter describes our design process throughout this project. This process involved three main stages: research, design, and then implementation and testing. First, we gathered information during the research stage so that we could make information decisions about our design options and individual design choices. As we explained in the report thus far, a portion of this research involved reviewing prior art to determine how similar systems have been designed and implemented. We were specifically concerned about the means to sense motion, capture images, process images, store images and to efficiently power these portions of the circuit. In addition, we researched methods for notifying the customer of newly sensed events and how we could implement them.

During the design phase, we chose components so that the final device would have all required functionalities. Based on the research, we chose sensors, a microcontroller, memory and a power supply to meet our requirements, a major concern being power efficiency. In addition, we chose components that we could interfaced together. Not only must they be able to communicate with each other, the prototype had to interface with the wireless communications portion of the overall system. Last, the group designed the user interface portion of this project to meet the expectations of the sponsor and for ease of development.

In the testing and implementation phase, we systematically implemented the design in pieces, and then implemented it as a wholly integrated system. Once parts arrived, we tested them individually and documented their performances to ensure that they worked and would perform adequately. Then, we configured the interfaces between components and modules. We also tested the interfaces to ensure communication was successful. Last, we combined subsystems and then the whole system and tested high-level functionality. All along this process, we documented our findings and evaluated our results. Figure 6 is a Gantt chart, a schedule detailing our individual tasks during this project.

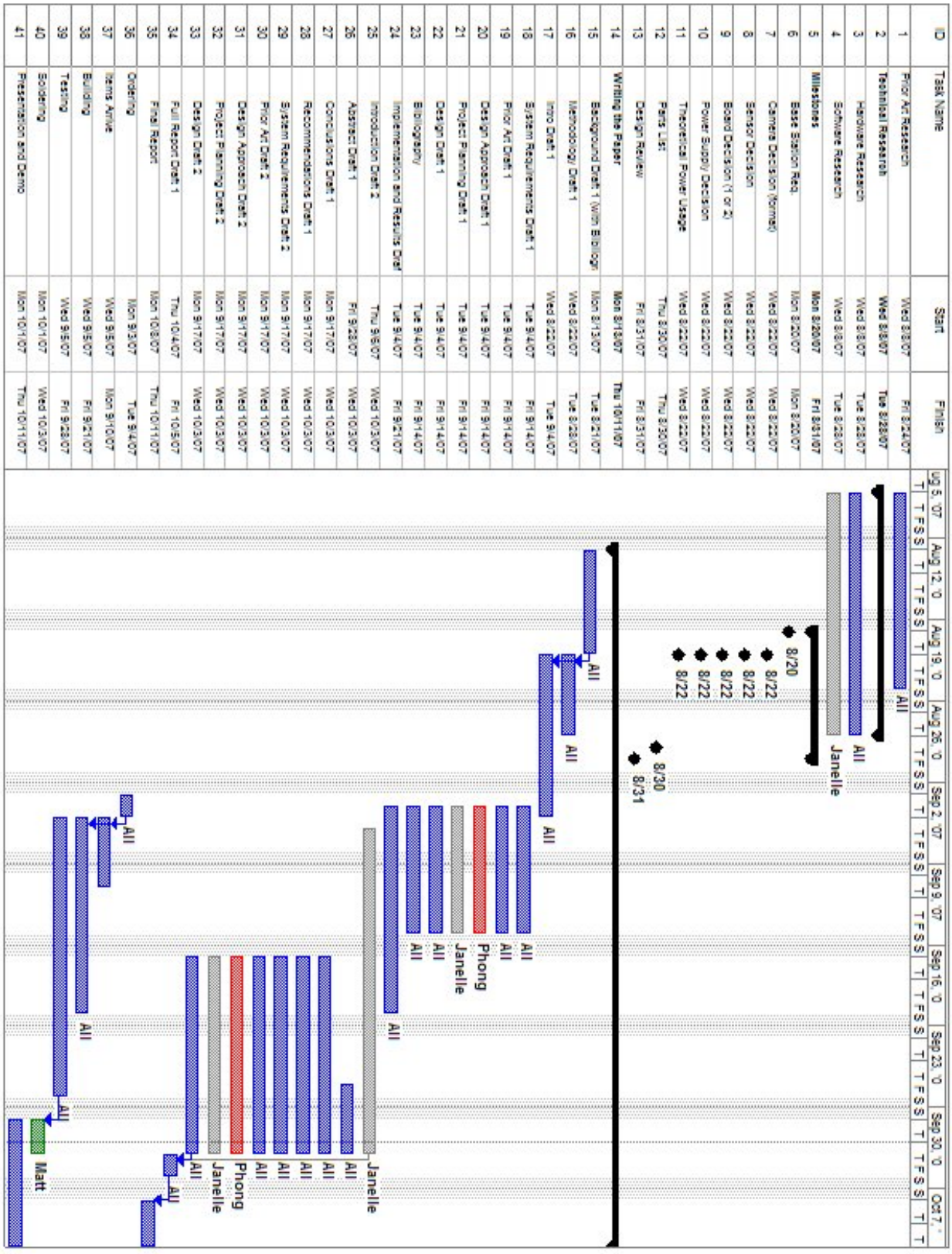


Figure 6: Project Schedule.

5 Design

In the following section, the we explain in detail the design which we concluded would meet our design goals and requirements for the camera node and base station of an efficient wireless security system. For each aspect of the design, the sections in this chapter contain relevant background information and our thought process behind our hardware and software decisions.

5.1 Motion Sensor

A major component of a security camera surveillance system is motion detection because it triggers the camera to either start recording or begin snapping photos. In this section, we describe our research about motion detecting methods and our selection of a motion sensor for this project.

5.1.1 Technical Research for Motion Sensor

This section explains some of the technical information behind motion detection that can be used for this security camera system. There are two main methods that were available for us to detect motion: radar and passive infrared. When radar motion detectors are active, they constantly sending out microwave radiation that bounces off objects in the surrounding area. When a signal returns to the detector, it compares the original signal to the reflected signal. If there is a difference between the two, the detector interprets it as a change in motion. Since the detector constantly generates microwave radiation, this means of motion sensing would not be very power efficient. As we were concerned about power, we did not consider radar motion detectors for this project. Additionally, as infrared motion sensors have become cheaper, it has been harder to find radar motion detectors. [19]

Infrared motion sensors work in a much different fashion than the radar motion detectors. Infrared energy can cause the atomic and subatomic particles of matter to have an increase in energy. As Riccardo Vanzetti stated in Practical Applications of Infrared Techniques, these sensors interpret the detected increase in energy as an increase in temperature. The change in temperature can then be determined by observing several different changes:

- Variation of physical qualities (volume, pressure, refractivity)
- Variation of chemical properties (infrared photography)
- Variation of electrical properties (conductivity, dielectric coefficient, secondary emission) [46]

The effect of infrared energy that was most interesting for motion detection in surveillance systems was that of the variation of electrical properties. To understand how the sensor works, one must study the physics behind the device.

The basis of this technology lies in the properties of certain dielectric compounds; these compounds, known as ferroelectric materials, can undergo spontaneous polarization in certain

situations. Depending on the temperature, the degree to which the materials polarize changes. As Vanzetti stated, when the materials are heated to the Curie Temperature, there is a major change in the crystalline structure of the ferroelectric materials. The Curie Temperature is the temperature at which ferromagnetic materials are no longer permanently magnetic [50]. At the Curie Temperature, the material loses its polarization and the dielectric coefficient changes greatly in value. It is for this reason that ferroelectric materials are used in pyroelectric detectors, which are the essential components of PIR sensors.

With the pyroelectric detectors, the concentration of the electric charge changes relative to temperature, as mentioned previously. When the dielectric compounds, typically tryglycine sulfate crystals, absorb infrared energy, a voltage appears between the poles of a capacitor proportional to the amount of infrared radiation [46]. This change in voltage represents a change in infrared radiation.

For infrared energy detectors to work for security purposes, they must be designed to be sensitive to the thermal body temperature of humans relative to the ambient temperature of the surroundings. When the detector senses this temperature, it will create a voltage [19].

5.1.2 PIR Sensor Options

This section presents some of the options for PIR sensors that are available for use in an application such as a security system. It is first important to describe the main criteria we considered when looking for PIR sensors.

Current output: The amount of current that the sensor can source is important as the output signal must be processed. Knowing the current waveform would help determine if the signal needs any conditioning before being processed by the microcontroller.

Sensing range: The sensing range is important since the motion detectors need to cover a potentially large area. With longer range capabilities of the sensor, fewer sensors would be needed to cover a region.

Sensing angles: Again, with larger vertical and horizontal sensing capabilities, the sensor would be able to cover a larger area.

Price: Price is important to monitor in all parts of a project in order to keep prototyping costs down and eventually the final cost of the product overall.

Table 3 summarizes several different PIR sensors that are possible options for motion detection in this project. Looking at Farnell and Digikey component suppliers, we found that the inventory was essentially the same, consisting of PIR sensors from Panasonic.

Table 3: Comparison of available PIR sensors.

Model	Current Output (μA)	Sensing Range (m)	Sensing Angles $\frac{hor}{vert}$(°)	Supply Voltage $\frac{max}{min}$(V)	Price ea. (€)
AMN11111	100	5	$\frac{100}{82}$	$\frac{6}{3}$	18.29
AMN12111	100	2	$\frac{91}{91}$	$\frac{6}{3}$	20.15
AMN14111	100	10	$\frac{110}{93}$	$\frac{6}{3}$	19.96
255-1811-ND	50	10	$\frac{110}{93}$	3	19.14

5.1.3 Selection of a PIR Sensor

After reviewing the selection of PIR sensors, we made a decision about which sensor would meet the specifications of this project. Since a radar motion detector did not seem like a viable option, we only considered passive infrared sensors. We found Panasonic's AMN14111 to be a suitable option. First, it has the longest range that we found—10m. It also has the widest angular detection ranges: 110° horizontal and 93° vertical. The downside to this, however, is that the actual detection zones for the detector are less dense than the detection zones of a sensor with a shorter range and smaller detection angles.

With multiple sensors in the car park, the likelihood that a thief or vandal would be able to avoid all detection zones, however, is very low. Given a sensor that has a denser X-Y cross-sectional coverage, the range would not be as large as 10m and therefore would not be able to detect across a lane in the car park. Generally, we found that the range of other PIR sensors were about 5 meters. This range would result in dead zones through the center of the lane, creating an area in which a criminal could avoid detection. Figure 7 shows the car park for which we are designing this surveillance system. We took some measurements of the car park and found that the average parking space is about 5m long. Therefore, a sensor with a 5m range would not be sufficient.

Figure 7: Foundation Car Park at the University of Limerick.

The Panasonic AMN14111 is also suitable since it is available with a digital output and has a built in amplifier so that it can be directly connected to a microcontroller. The current consumption of the device is typically only about 170μA, which means that it would consume a relatively small quantity of power. This PIR sensor is also simple to use since it has only three pins: supply, ground, and output. Figure 8 shows the selected sensor.

Figure 8: Picture of PIR Sensor.

5.2 Camera Module

This section explores the major parts of the camera module: the CMOS image sensor, the image processor, and the JPEG compression engine. Also in this section, we describe some of the possible options to fulfill this module's functions and which one we chose.

5.2.1 Technical Research for Camera Module

As just noted, the camera module consists of three major components: the CMOS image sensor, the image processor, and the JPEG compression engine. A CMOS sensor is a silicon chip that consists of many photosensitive diodes. These diodes, called photosites, are capable of capturing light and would capture the information for individual pixels of an image. When the photosites are exposed to light, they record the brightness of the light. This detection is accomplished through the accumulation of photons. In a light area, there will be many photons collected and few photons in a darker area. When the photosites are no longer exposed to light, the number of photons for each site is converted into a digital number that represents a color. All of these the colors put together for each pixel create a picture. [15]

The next major part of the camera module consists of the image processor. This component conditions an image. It is able to perform lens shading and corrects defective pixels. It can also have automatic exposure control and automatic gain control. Furthermore, the image processor can reduce noise in the image and make edges more distinct. Finally, the image processor can control contrast and brightness of the image to give an image better clarity. Once the image processor finishes processing the data, it transmits the data to the next major component of camera module [32].

The final component is the JPEG compression engine. This component decreases the image file size. There are several common file formats for images; some examples of their extension are .bmp, .gif, .png, and .tiff. The file format with the greatest amount of potential compression is JPEG, named after the Joint Photographic Experts Group. This format compresses an image file by consolidating repeating data. With greater compression, however, the image loses quality and clarity. Depending on the image and the desired quality of the output, the compression engine can reduce the amount of data in a file by a factor of twenty or greater [20]. By using a JPEG compression engine, we sought to reduce the amount of storage necessary for image files and to reduce the time required to wirelessly transmit these images, therefore reducing overall power consumption.

5.2.2 Camera Module Options

This section describes different available camera modules and how we evaluated them. The following lists the criteria we used to determine if a module would meet the requirements for this project.

Frame rate: The frame rate is generally given in terms of number of frames per second that a camera can capture. One of the customer requirements was that the camera takes two pictures per second for five seconds upon the system detecting motion. A camera with a frame rate that cannot be adjusted to such a value is undesirable; too many images may have to be transferred wirelessly or too little information may be captured.

Power Consumption: As we sought to minimize power consumption for the whole system, this was an important number. The less power the camera consumes while awake, the longer the system's power supply will last. As we did not know what typical power consumption for cameras was before beginning our research, we sought to find the minimum value here. Generally, we sought a camera that drew less than 100mA.

Package: The package of a camera is relevant as some packages are easier to implement than others. As we had limited time and limited resources, we sought a camera that met the requirements and that could be implemented with the resources available to us.

Output format: Since wireless communication between nodes is necessary for this project, we sought either a compressed digital format, such as JPEG.

After defining the criteria, we gathered information about camera modules in the market. We chose a set of camera modules to represent what was available. Table 4 describes those camera modules and their specifications.

Table 4: Camera module options.

Manufacturer	Transchip	ST	Electronics123	Electronics123
Model	TC6030	VS6724	C628	C238-7640
Frame Rate	Programmable 1-15 fps			0.75-6 fps/still images
Power Consumption	150mW	256mW	1.25W	198mW
Package	24 pin camera module		PCB	PCB
Output Format	YUV and RGB	M-JPEG	JPEG	JPEG

The modules described in Table 4 contain not only cameras on-board, but also processors that can be programmed to alter the different functions and capabilities of each device. The first type of camera we considered was intended for mobile devices. These cameras were often accompanied with many features. Transchip's TC6030 has a programmable frame rate of 1-15 frames per second, which means the desired rate of two pictures per second is achievable. It also consumes relatively little power, comes with a board to board connector, and has an onboard analog-to-digital converter. Additionally, it had a sleep mode, fully programmable signal processor, and a snapshot mode. The product specification sheet, however, did not specify what type of programming was necessary to utilize all of the modules features [4]. When we emailed Transchip about the module to gain a better understanding of the product and the possibility of using it for this project, they failed to reply.

We also considered a camera module produced by ST. It is also intended for mobile devices. Though it was not specifically listed in the specification sheet, it stated that the frame rate is programmable and that the camera is capable of taking still images. With 256mW power

consumption, this camera uses the second highest amount of power of the ones we reviewed, though not by a significant amount. The camera is mounted on a board and needs a connector to operate, though there is no information on programming its processor. When in standby mode, the camera draws very low current; it is more power efficient than the TransChip camera. Finally, the camera is capable of outputting images in a JPEG format [49].

The C628 camera module was the first module we considered from Electronics123. One can program the camera to take snapshot images or capture video. While the frames per second is not specified, it is alterable. Drawing 250 mA when active, this camera uses 1.25 Watts of power, a relatively large amount of power. The modules consists of two PCB boards, with the lens and CMOS sensor mounted on a separate PCB. The camera has many extra features, such as time and date stamping and file management. An attractive feature is its JPEG compression engine. One can control this module with an external microcontroller [11].

The final module that we reviewed was the C328-7640. The module is capable of taking snapshot images, in addition to capturing video. Its frame rate is adjustable - 0.75 to 6 frames per second. When in operation the camera draws only 60mA of current and needs only run on a minimum of 3V. Its sleep mode allows the module to be more power efficient. A processor onboard compresses the data into the JPEG format and the quality of the images can be adjusted. By using an external microcontroller, like the above C628, the camera can be programmed to perform a number of different functions [11].

5.2.3 Selection of a Camera Module

We chose the C328-7640 camera module for this surveillance system prototype. Of the camera modules reviewed, this camera module came closest to meeting our design goals.

The module not only has an analog to digital converter, but also JPEG compression, which is necessary for efficient wireless transmission of the images. The quality of the JPEG images can also be adjusted, so we can customize the amount of image quality sacrificed to reduce transmission time. The camera is fairly low power, drawing only 60 mA, which is important for achieving overall system efficiency. While the C628 has more features, it draws four times the current that the C328 draws. The C328 needs only a 3V supply, while the C628 needs a 5V supply. As mentioned previously, by using an external microcontroller, we can program the camera to perform the functions this project requires. These functions of the camera are outlined in Table 5.

Table 5: List of commands for camera module.
http://www.electronics123.net/amazon/datasheet/C328-7640UM_V3.pdf

Command	**ID Number**	**Paramter1**	**Parameter2**	**Parameter3**	**Parameter4**
Initial	AA01h	00h	Color Type	RAW Resolution (Still image only)	JPEG Resolution
Get Picture	AA04h	Picture Type	00h	00h	00h
Snapshot	AA05h	Snapshot Type	Skip Frame Low Byte	Skip Frame High Byte	00h
Set Package Size	AAo6h	08h	Package Size Low Byte	Package Size High Byte	00h
Set Baud-rate	AA07h	1st Divider	2nd Divider	00h	00h
Reset	AA08h	Reset Type	00h	00h	xxh
Power Off	AA09h	00h	00h	00h	00h
Data	AA0Ah	Data Type	Length Byte 0	Length Byte 1	Length Byte 2
SYNC	AA0Dh	00h	00h	00h	00h
ACK	AA0Fh	00h	NAK counter	Error Number	00h
Light Frequency	AA13h	Frequency Type	00h	00h	00h

For programming or communications, the camera module uses an external interface - a UART port. This port has four pins: power, ground, a transmit line, and receive line. The interface allows the camera module to communicate with any microcontroller that has a UART interface.

We did not consider the cameras intended mobile device as we could not gather information about programming them, even though they are relatively power efficient and run off of low supply voltages. They also need many more interface pins than the C328 which only has four. Therefore, we decided the C328 would be easier program. The user manual, which we accessed on the manufacturer's website, has an abundance of information about programming the camera, as well as sample code. The C328 had a combination of features, low power consumption, and we felt we would be able to successfully program it; therefore we chose this module for our security camera system.

5.3 Development Board

This section describes the central processing unit of the camera node - the development board. We first give an overview of microcontrollers and then present the available options for development boards. The last part of this section describes our decision to use one of these boards in the project.

5.3.1 Microcontroller Research

The main element of a development board is the microcontroller, which is commonly used in numerous electronic devices. A microcontroller is a single integrated chip that typically consists of a central processing unit (CPU), volatile and non-volatile memory, timers, temperature sensor, and analog-to-digital converters. Microcontrollers are utilized for design-optimization and system-effectiveness.

In this particular project, the microcontroller of the camera node served as a central unit that was responsible for several tasks: handling the signal from the motion sensor, obtaining the image from the camera module, transferring the image to the external storage, and controlling the operation of the transceiver module. Some of the popular lines of microcontrollers that could perform these tasks include the ARM and AVR microcontrollers from Atmel Corporation, MSP430 microcontroller from Texas Instruments (TI), and PIC microcontroller from Microchip Technology.

5.3.2 Microcontroller Options

To minimize the time necessary to become familiar with a microcontroller, we preferred to use one with which we had experience. Since we had never used the ARM, AVR or PIC microcontrollers, we only considered the MSP430 microcontrollers. The MSP430 is well-known for its ultra-low power consumption. Furthermore, TI provides excellent resources and technical support for its MSP430 line of microcontrollers. There are four general families of MSP430 microcontrollers: MSP430x1xx, MSP430x2xx, MSP430x3xx, and MSP430x4xx family. Each family has its own features. We decided to employ the MSP430x1xx, which is the basic family. There are over thirty different versions of these chips in this family, however, so our next step was to select a chip that was suitable for this project.

Before making a decision, we considered the environment in which we would put a microcontroller. The camera node consisted of a camera module, a motion sensor, memory, a transceiver, and the development board. The development board had to be able to interface with those other devices in the camera node. As we had chosen the camera module, we knew the MSP430 chip must have a UART port. Also, the chip needed to have an interrupt port to connect with the motion sensor. Finally, the chip needed additional UART ports or Serial Peripheral Interface (SPI) ports to communicate with the memory and the transceiver. With these requirements, we removed two thirds of the MSP430x1xx family as an option.

All of the remaining chips had similar features, but some had more integrated peripherals than others. In addition, the size of the flash programming memory of those chips ranged from 16KB to 60KB, and the size of the on-board random access memory (RAM) ranged from 512 bytes to 10KB. Another consideration for choosing a microcontroller was the availability of development boards. We chose Olimex Limited, a prototyping board manufacturer, as a supplier because they offered reasonable prices for a variety of boards. This supplier manufactured several boards which incorporate the MSP430x1xx family of microcontrollers. Only three boards that

contained the MSP430F149, MSP43F169, and MSP43F1611 microcontrollers would satisfy the requirements. Table 6 summarizes the specifications of these three development boards from the MicroController Pros Corporation distributor.

Table 6: Three types of MSP430x1xx microcontroller development boards.

Board Description	Flash	RAM	Other Integrated Peripherals	Interface	Price
MSP430F149 Prototype Board, RS232	60kB	2kB	• analog comparator • hardware multiplier	two USART (SPI and UART)	€24.67
MSP430F169 Prototype Board, RS232	60kB	2kB	• analog comparator • hardware multiplier • two 12-bit DACs • DMA controller • SVS	two USART (SPI, UART or I2C)	€29.62
MSP430F1611 Prototype Board, RS232	48kB	10kB	• analog comparator • hardware multiplier • two 12-bit DACs • DMA controller • SVS	two USART (SPI, UART or I2C)	€32.44

5.3.3 Selection of a Microcontroller

After gathering information about available development boards with chips from the MSP430x1xx family, we evaluated these options. As one can see in Table 6, all prices were similar. The last two chips had more integrated peripherals than the first one; however, those peripherals were not a main concern. The next noticeable difference is the RAM size of the chips. The microcontroller would be responsible for processing the image data from the camera module. Since the image size can vary from 2KB to 200KB, the camera module would transfer many data packages to the microcontroller. Therefore, it would be efficient to have the on-board RAM be as large as possible to buffer more data before storing in permanent memory. Although the MSP430F1611 chip has a smaller amount of flash memory than the other two, it has the largest RAM and would serve the purposes of this project. Figure 9 shows the chip we selected, as well as the components on the development board.

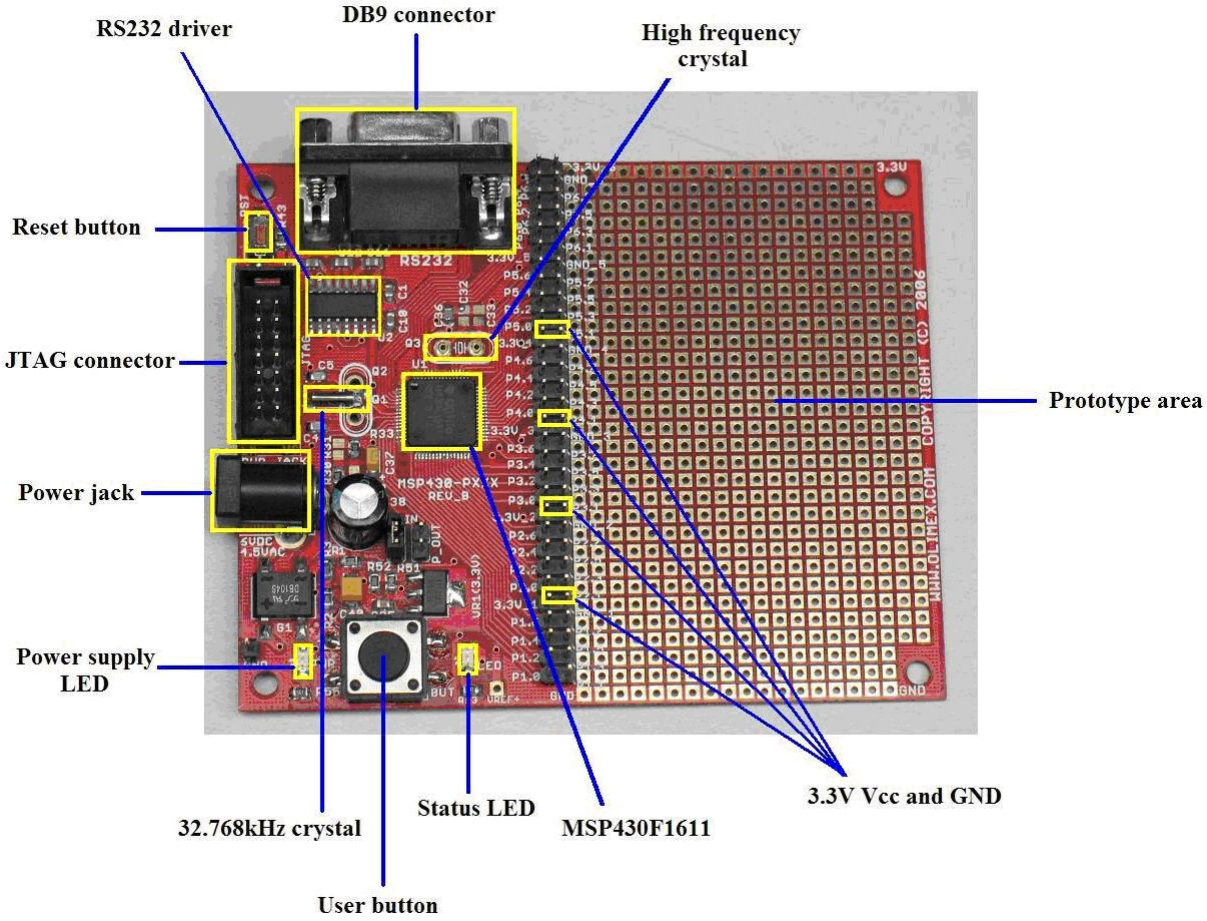


Figure 9: MSP430F1611 development board.

The MSP430F1611 microcontroller on this board had an operating voltage of 3.3V, with its own voltage regulator. Users could program the microcontroller via a JTAG connector by using IAR Embedded Workbench, an integrated development environment for building and debugging embedded applications. The board had pre-built RS232 driver which one could use to communicate with the camera module. The status LED would help a programmer with testing peripherals or monitoring the system. The extension headers would give the user easy access to the pins of the microcontrollers for connecting interfaces. Furthermore, a user could power external peripherals by using the on-board voltage supply and ground bus.

We considered the power consumption of this board in regards to overall power efficiency. After reviewing the board's schematic provided by Olimex, we identified the following components that would draw current from the power supply when in use: the MSP430F1611 microcontroller, the LM1117 linear regulator, the ST3232CD RS232 driver and receiver, and the LEDs; however, we could disable the status LED. By studying the datasheet of the previously listed components, we were able to obtain the necessary information to estimate the current consumption of the board. We used this data later in the report to calculate the overall power consumption of the camera node, regarding the power supply requirements. We summarized these estimations in

Table 7 and Table 8, the former describing the normal operating more and the latter describing the low-power mode.

Table 7: Current consumption of the development board in operation mode.

	Typ.	Max.	Unit
MSP430F1611	4.0	4.8	mA
LM1117	5.0	10.0	mA
ST3232CD	0.3	1.0	mA
Power LED	4.5	5.0	mA
Status LED	4.5	5.0	mA
Total current	18.3	25.8	mA

Table 8: Current consumption of the development board in low-power mode.

	Typ.	Max.	Unit
MSP430F1611	0.0002	0.095	mA
LM1117	5.0	10.0	mA
ST3232CD	0.3	1.0	mA
Power LED	4.5	5.0	mA
Status LED	-	-	mA
Total current	9.8002	16.095	mA

5.4 Memory

This section describes our process for choosing memory for the camera node. It begins with a brief overview of memory technologies. We then explained the types of memory that could be used in this design. Finally, we elaborated on our selection of memory for the camera node.

5.4.1 Technical Research for Memory

There are two types of memory technologies: volatile and non-volatile. The first type, volatile memory, requires power to maintain the stored data. As soon as the supply source stops providing power, volatile memory's data would be lost and could not be retrieved. Two common forms of volatile memory are static random access memory (SRAM) and dynamic random access memory (DRAM). SRAM and DRAM both have their strengths and weaknesses. SRAM can hold data without the need of external refreshing, as long as the power supply is available. In contrast, DRAM must be refreshed frequently in order to hold its stored information. We took into account other characteristics of volatile memory such as access time, capacity, and cost when considering this type of memory. [6, 17, 42, 48]

The second type of memory is non-volatile; it retains stored information without a power supply. Two common forms of non-volatile memory are read-only memory (ROM) and flash memory. ROM is most commonly used to store firmware since it cannot be rewritten. Therefore, ROM provides reliability and security against accidental or intentional changes to its contents. There are several types of ROMs such as Programmable ROM (PROM), Erasable Programmable ROM (EPROM), and Electrical Erasable Programmable ROM (EEPROM). The major difference between PROM and EPROM or EEPROM is that PROM is one-time programmable device, whereas EPROM and EEPROM can be erased and reprogrammed under software control. Another common form of non-volatile memory is flash memory which is a specific type of EEPROM. Flash memory is block-wise writable, while EEPROM is byte-wise writeable. One advantage of flash memory is that its capacity ranges widely from a few hundred megabytes to several gigabytes. Furthermore, flash memory presently can achieve a transfer rate of 12 MB/sec. Therefore, flash memory is extensively used for general storage and transfer of data among digital devices such as computers, digital cameras, and mobile phones. [27, 31]

5.4.2 Memory Options

Compatibility was our first concern when gathering our memory options. Since the MSP430 microcontrollers do not have a memory bus, they could not interface with memory that requires an address bus. Hence, the only memory type that could be compatible with our system was the removable flash memory cards. Fortunately, there was a variety of removable flash memory formats available. Some of these included CompactFlash (CF), xD-Picture card, SmartMedia, Memory Stick, Security Digital (SD), MicroSD, MiniSD, Pen Drive, and MultiMediaCard (MMC). These memory formats have different maximum capacity, physical weight and size, data transfer rate, and number of pins.

After our research, we only found available documentation regarding the interface between the MSP430 microcontroller and the SD card. Fortunately, the SD card is readily available in the market for a reasonable price. Moreover, this popular memory format has been proven to be reliable data storage. Thus, we decided to use an SD card for storing images from the camera module. Table 9 presents the available SD cards from some popular distributors.

Table 9: SD memory card comparison.

Description	Part Number	Manufacturer	Distributor	Capacity	Price
SD 80x Speed	10449	Transcend	MemoryC	128MB	€7.95
SD 80x Speed	10438	Transcend	MemoryC	256MB	€8.95
SD Card	5014554	ScanDisk	Farnell	256MB	€70.31
SD 80x Speed	10307	Transcend	MemoryC	512MB	€11.00
SD	SDM512-ND	Toshiba	Digikey	512MB	€45.77

5.4.3 Selection of a Memory Card

The two features of concern about the SD cards were the capacity and the price. As described in Table 9, one of the Transcend cards had the smallest capacity of 128MB. With a rough estimation, the card could hold up to 65000 images that has an average size of 2KB, or 655 images that had an average size of 200KB. This card has the lowest cost. As this capacity is sufficient for our proof-of-concept project, we selected the 128MB Transcend Secure Digital 80x Speed card for image storage.

The operating voltage of the selected memory card is between 2.7V and 3.6V. An attractive feature of the card was its data transfer rate - up to 12MB/sec. This rate was about 70% faster than the average SD card [44]. The most notable aspect of this type of memory was that we were certain it could interface with the MSP430 microcontroller. Specifically, the selected MSP430F1611 microcontroller and the Transcend card could communicate through the SPI port. We described the details of this interface in the Implementation and Results chapter.

5.5 Power Source

This section describes various types of battery technology and provides an overview of the available options for each battery type. Finally, we the calculated power consumption of the camera node system to determine the required specifications of the battery for our design.

5.5.1 Battery Technology Research

One requirement for this surveillance system project was that the camera nodes should be able to run on battery power for at least a year without any maintenance. Since we had to use a wireless method of power, we examined different types of primary (not rechargeable) and rechargeable batteries; some examples are lead acid, alkaline, nickel cadmium, nickel metal hydride, lithium and lithium ion batteries. In the remainder of this section, we list various types of battery technologies and describe the key features.

The *Lead acid* battery is the oldest type of rechargeable battery on the market. The main chemical element inside the lead acid batteries is lead. As a result, this type of battery is considered environmentally unfriendly; however, this type of battery is still common. One advantage is that the lead acid batteries provide reliable service and require low maintenance when being used correctly. In addition, they are inexpensive and best suited for power applications in which weight is not a concern [8].

Alkaline batteries have been used extensively to power numerous electronic devices. They make use of the chemical reaction between zinc and manganese dioxide. The capacity of alkaline batteries strongly depends on the load. AAA-sized alkaline battery might have an effective capacity of several thousands milli-Ampere hours (mAh) with a very low current draw. But, if the load draws 1000mA, the capacity can be as little as 700mAh [5]. These alkaline batteries are often primary batteries. Although there are alkaline rechargeable batteries, they are uncommon because people generally prefer to purchase the new, more inexpensive primary batteries.

The *Nickel Cadmium* (Ni-Cd) battery is a popular type of rechargeable battery. Ni-Cd batteries use nickel hydroxide as the positive electrode and cadmium as the negative electrode [29]. Ni-Cd batteries provide a constant voltage throughout their service life, and they are best suited for high current applications.

The *Nickel Metal Hydride* (Ni-MH) battery is a type of rechargeable battery similar to a Nickel Cadmium (Ni-Cd) battery. Ni-MH batteries are manufactured with nickel hydroxide for the positive electrode and hydrogen-absorbing alloys for the negative electrode [30]. Ni-MH batteries have become predominant over Ni-Cd batteries because of their high capacity.

Lithium batteries use lithium metal or lithium compounds for the positive electrode. The negative electrode is manufactured with various chemical compounds. Depending on the chemical compounds used, lithium batteries can produce a nominal voltage ranging from 1.5V to 3.8V [24]. Presently, lithium batteries are widely used in industry.

The *Lithium Ion* (Li-on) battery is a type of lithium battery that is rechargeable. Lithium ion has been the fastest growing and most promising battery chemistry. A deterrent from this type of battery is the necessity for a protection circuit to maintain safe operation. Because of its lightness and high energy density, Li-on batteries are excellent for portable electronics devices, such as notebook computers and mobile phones [24].

5.5.2 Battery Options

All the types of battery we described in the Battery Technology section have both advantages and limitations. In order to determine a suitable battery type, we established the following criteria:

- Low cost - the price should be less than €20.00
- Rechargeable
- Simple implementation
- High capacity
- High nominal voltage—the battery nominal voltage should at least 4.5V

The above concerns are listed in a decreasing order of importance. We suggested a nominal voltage of at least 4.5V because the development board requires an input voltage between 4.5V and 6.0V. In addition, the rechargeable capability was necessary as no primary battery could maintain the operation of the camera node for a year. Specifically, the transceiver module on the camera node required a continuous power supply; thus, it would always draw current with a typical value of 36mA. Additionally, the development board consumes 9.8mA even in the sleep mode. In the most power efficient situation, the camera node would be entirely inactive as there would be no activity of any kind, nonetheless criminal activity, to detect in the car park. But, the system would still consume a total of 45.8mA. The common largest battery capacity that

we found in the market was 27Ah. A simple calculation would give us the number of operating hours as follows:

$$\frac{2700mAh}{45.8mA} = 589.5h$$

If the system operated eight hours per day, the resulting number of operating hours from this calculation was approximately seventy-four days. As a result of this calculation, we concluded that it was impractical to use primary batteries to power the camera node when the requirement demands at least a year of service from the battery. Thus, this project required a rechargeable battery. Based on our criteria for a suitable battery, we created Table 10 to compare the described battery types.

Table 10: Battery Types Comparison.

Battery	**Cost**	**Capacity**	**Implementation**	**Voltage**	**Rechargeable?**
Lead acid	Inexpensive	Varied	Simple	High	Yes
Alkaline	Inexpensive	Large	Simple	Low	Mostly No
Ni-Cd	Moderate	Varied	Simple	Low	Yes
Ni-MH	Moderate	Varied	Simple	Low	Yes
Lithium	Expensive	Varied	Protection circuit	Moderate	No
Li-on	Expensive	Varied	Protection circuit	Moderate	Yes

We did not consider alkaline and lithium batteries, as they are not rechargeable. The Li-on battery requires protection circuit which creates complexity for the design. Moreover, they are expensive; hence, we eliminated this option. Both Ni-Cd and Ni-MH batteries have reasonable cost, however, the nominal cell voltage of these two types is typically 1.2V which is substantially lower than the desired value 4.5V. We decided to use lead acid batteries, as they appeared to meet our criteria.

5.5.3 Selection of Battery

Before determining a suitable lead acid battery for this project, we estimated the power consumption of the system. The camera node should operate in two modes: sleep mode and active mode. In the sleep mode, the camera module and the microcontroller consume insignificant power since they would not be fully operational. In the active mode, all the components of the camera node should operate at their typical working conditions. Table 11 and Table 12 summarize the data we gathered about the theoretical current consumption for each mode.

Table 11: Current consumption in sleep mode.

	Min.	Typ.	Max.	Unit
PIR sensor	0.15	0.25	0.30	mA
Transciever module	36.00	36.00	42.00	mA
Development board	9.80	9.80	16.10	mA
Camera module	0.10	0.10	0.10	mA
Memory	0.20	0.20	0.20	mA
Total current consumption	46.25	**26.35**	58.70	mA

Table 12: Current consumption in active mode.

	Min.	Typ.	Max.	Unit
PIR sensor	0.15	0.25	0.30	mA
Transciever module	36.00	36.00	42.00	mA
Development board	18.30	18.30	25.80	mA
Camera module	60.00	60.00	60.00	mA
Memory	40.00	50.00	50.00	mA
Total current consumption	154.45	**164.55**	178.10	mA

We expect the system to operate eight hours - from 10:00pm until 6:00am. We examined three scenarios: the worst scenario, normal operation, and the best scenario. We calculated the required battery capacity for one day operation in each scenario. Table 13 presents these calculations.

Table 13: Required capacity for one day.

	Min.	Typ.	Max.	Units
Worst case	1236	1316	**1425**	mAh
Best case	370	371	470	mAh
99% sleep mode, 1% active mode	379	380	479	mAh
50% sleep mode, 50% active mode	803	844	947	mAh

In the worst scenario, we found that the battery would need a capacity of 1425mAh. The majority of the lead acid batteries in the market today have a much greater capacity than the worst-scenario capacity. Table 14 shows several choices of the lead acid batteries.

Table 14: Available 6V lead acid batteries.

Manufacturer	**Distributor**	**Part Number**	**Capacity (mAh)**	**Weight (g)**	**Price ea.**
Camden Electronics	Farnell	BEG060120	12000	2000	€17.28
Yuasa	Farnell	NP12-6	12000	2050	€29.05
Sonnenchein	Farnell	A506/10.0S	10000	2050	€50.72
Silicon Solar Inc.	Silicon Solar Inc.	16525	8500	1493	€12.00
Cyclone	Farnell	9859-0012	8000	1430	€25.31
Yuasa	Farnell	NP7-6	7000	1280	€18.95

As one can see in Table 14, all the selected batteries have a nominal voltage of 6.0V. We ruled out the batteries with a price over €20.00. Furthermore, we also eliminated the choices of batteries that are manufactured by Camden Electronics and Yuasa because those batteries cost three times as much as the price of the battery from Silicon Solar Inc. As a result, we selected the battery sold by Silicon Solar Inc. The chosen battery has a large capacity of 8500mAh; in the worst case scenario, this capacity could maintain the system's operation for approximately 4 days without being recharged. Figure 10 shows the battery that we chose for this project.

Figure 10: 6V sealed lead acid battery.

5.6 Charging Method

This section discusses the methods for charging the selected lead acid battery. It also describes charge controllers, which often accompany the charging devices. After that, we presented the available options of battery charging subsystems. Finally, we explained our decision for the devices that would suit this project.

5.6.1 Technical Research for Charging Method

There were two methods that we considered for recharging the batteries: wind turbines and solar panels. By using a charging method, the user would not necessarily need to maintain the battery for at least a year, per the project requirements. We ruled out a couple charging methods initially, such as AC charging and using a generator. While the batteries could be recharged with AC power, this method would require someone to remove the batteries from the system to charge. Using a gas generator to power the system would not be practical since each node would need to be wired to the generator; running wires opposes our sponsor's objective for this system to be wireless. Also, someone would need to replace the gas in the generator, which would not allow the system to be maintenance free for a year.

At first glance, wind power seemed like it would be a viable option to help power the security system. Many wind turbines are designed to charge batteries and are accompanied with charging regulators to prevent overcharge from occurring. In addition, the regulators are designed to work with each specific turbine. The Mayo Energy Agency points out that small wind turbines only become feasible solutions for power when they are in a remote area far from utility lines. By their standards, a small wind turbine is one that can cover the electricity needs of the average Irish home. For these wind turbines to be a good source of energy, there must be an ample amount of wind and the turbine rotor diameter must be an estimated 5m. In terms of pricing, the windmill would cost about €6,350 as a part of a system that costs €19,050 [41].

Solar panels are also available as an option, providing the possibility of charging the battery everyday. Solar panels capture the energy that the sun radiates and converts it into usable energy. This process is accomplished with semiconductors that make up the solar cells of the solar panel. They convert the sunlight into electricity through the photovoltaic effect. This effect describes the event during which the energy of light mobilizes charged particles in the semiconductors and then separates them to produce current [26]. Custom sizes for solar panels are available, allowing one to choose an appropriate power rating. There are also many prefabricated solar panels with standard outputs.

A device that commonly accompanies a charging device is a charge controller. The purpose of the charge controller is to condition the output of the charger so that it can safely charge lead acid batteries. It also prevents the battery from overcharging and from discharging through the solar panel. While it is possible to build a charge controller using analog components, it is more practical to buy a charge controller that is already built. The system is fairly simple to set up. The positive and negative leads of the solar panel connect to the respective terminals on the

charge controller; the positive and negative terminals on the battery connect to the respective terminals on the charge controller, and finally, the positive rail and ground for the load connect to the respective terminals on the charge controller.

5.6.2 Device Options for Charging Method

This section describes options for devices that can power the camera nodes. Wind power is not a viable option for several reasons, although the price was enough of a drawback to keep us from considering this option any further. Others included space requirements and installation efforts. We then proceeded with using solar power as a means of charging the batteries as it was applicable to our project's needs. Silicon Solar Incorporated sells many different models with varying voltages, output power, and sizes. Table 15 describes some of their options.

Table 15: Silicon Solar Inc solar panel options.

Solar Panel Model	**Voltage** (V)	**Output Power** (W)	**Size** (cm)	**Price**
04-10775WH	12	5	31.8x31.8x2.5	€39.18
SKU9358	12	7	15.2x35.6x2.5 folded, 30.5x35.6 unfolded	€53.43
SKU2238	6	4	15.2x35.6x2.5	€28.48
SKU8473	6	7	30.5x35.6 7	€52.00
SKU: 05-1200C	15.4	1.54	27x17.5	€39.18

As Table 15 portrays, there are many different solar panels available. While the table is limited to several possible choices that would be suitable for this project, there are many more panels available in the marketplace that have different operating voltages, output power, and sizes. We limited the review of panels to lower voltages because of the voltage rating of the battery we chose. The constraint on the charge controller is that it must match the voltage of the solar panel and the battery that it is charging and that it must be able to handle more current than the panel can source. Silicon Solar Incorporated sells several different charge controllers that are compatible with their solar panels. Table 16 summarizes different available charge controllers and some of their specifications.

Table 16: Silicon Solar Inc charge controller options.

	Voltage (V)	**Price**
06-1024	12	€24.92
16669	12	€80.52
SKU17849	6	€7.09

When we chose the charge controller, we considered that the voltage rating must match that of the battery and that the current rating must be greater the current the solar panel can source. With the new digital technology, charge controllers can utilize microcontrollers for greater accuracy and better charge protection. This technology comes with a much higher price than a simpler analog controller, such as the 6V device in Table 16.

5.6.3 Selection for Charging Method

This section presents our method for choosing the solar panel and charge controller. Since we chose a 6V battery, the choice for a compatible solar panel was smaller. We then determined the necessary power rating for the solar panel. To minimize cost, we determined if the smaller of the two available 6V solar panels (model SKU2238) would be sufficient for the power needs of this project. With a size of 0.1524m by 0.3556m, the panel has an area of $0.0542m^2$. We then decided to find solar radiation data for Limerick. The amount of solar radiation would give us an idea of the amount of energy that is available for the solar panels to collect. We chose to use the month with the least amount of energy available; if the panels can charge the battery during the months with the least amount of available solar radiation, they should be able to charge the batteries in all months. Figure 11 illustrates the number of hours of sunshine per month. The data in Table 17, obtained from NASA and collected for Limerick, Ireland, shows solar energy information for each month.

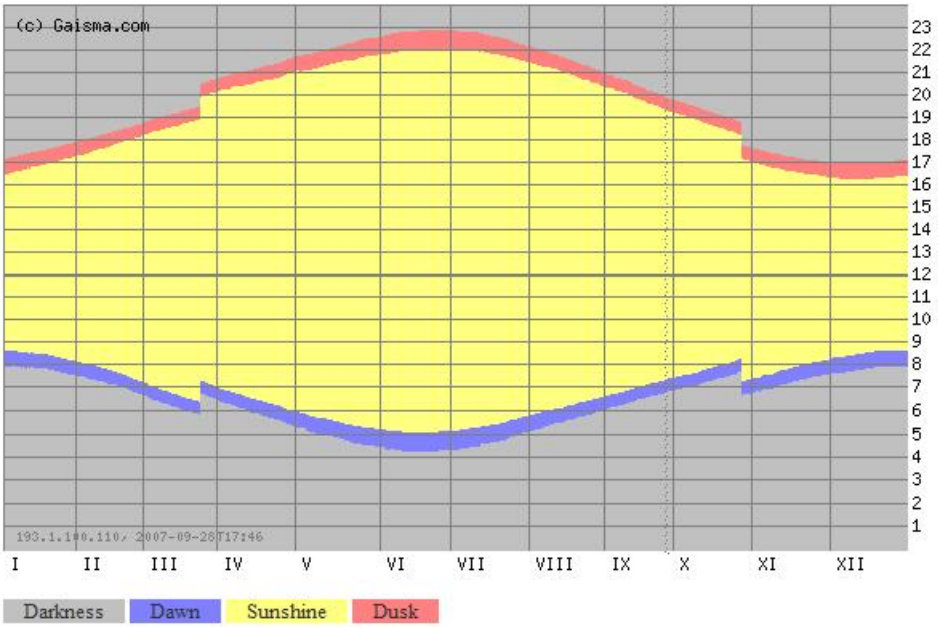


Figure 11: Sunrise, sunset, dawn and dusk times for Limerick Ireland. http://www.gaisma.com/en/location/limerick.html

Table 17: Solar energy and surface meteorology for Limerick, Ireland. http://www.gaisma.com/en/location/limerick.html

Variable	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII
Insolation, kWh/m²/day	0.64	1.10	2.03	3.43	4.43	4.47	4.19	3.64	2.61	1.37	0.71	0.46
Clearness, 0 - 1	0.33	0.32	0.35	0.41	0.42	0.39	0.38	0.40	0.38	0.32	0.29	0.29
Temperature, °C	4.94	5.01	6.44	7.78	10.82	13.42	15.68	15.72	13.62	10.30	7.26	5.62
Wind speed, m/s	10.07	9.45	9.14	7.70	7.01	6.33	6.34	6.64	7.45	8.48	8.97	9.62

From Figure 11, one can see that December is the month with the least amount of solar radiation. In Table 17, the value of interest to us in this chart was insolation. According to GAISMA, insolation is "'The monthly average amount of the total solar radiation incident on a horizontal surface at the surface of the earth for a given month.'" [23] Again, December has the lowest value. The insolation for December is:

$$0.46 \frac{kWh}{\frac{m^2}{day}}$$

Using this value, we then determined for the size of our solar panel how many ampere-hours (Ah) the battery can be charged. These calculations represent the best case for the average day

in the worst case month; though it would be better to consider the typical or worst case, we used the most precise numbers available to us. We performed the calculations mainly to determine whether or not it would be plausible to use the solar charger and to justify purchasing it.
To determine the Ah for one day, we multiplied m^2/day by one day to make a cancellation.

$$\frac{\frac{0.46kWh}{m^2}}{day} \cdot 1day = \frac{0.46kWh}{m^2}$$

We then determined the total watt-hours (Wh) possible for the size of the smaller solar panel we were considering, which was 15.2cm x 35.6cm, or 0.05419m^2. We multiplied the resulting value from the previous calculation to cancel m^2 and.

$$\frac{0.46kWh}{m^2} \cdot 0.05419m^2 = 0.02493kWh$$

$$0.02493kWh \cdot 1000 = 24.93Wh$$

The next calculation involved using the power equation $P = IV$ to determine the total number of Ah that the battery could charge in a day. Since the solar panel of interest is rated for 6V, we used this number in the calculation, though we know that the actual output may vary.

$$\frac{P}{V} = I$$

$$\frac{24.93Wh}{6V} = 4.155Ah$$

This calculation shows the total possible Ah that a solar panel of the given size can charge a battery in one day. We then used a power calculation again to figure out how much current the panel can source. The value that we calculated represents a best case situation. We were not able to use a typical value since Silicon Solar does not supply specification sheets with their solar panels.

$$\frac{P}{V} = I$$

$$\frac{4W}{6V} = 0.6667A$$

While sourcing the maximum amount of current, we determined how long it would take the solar panel to supply a battery with 4.155 Ah—the calculated maximum Ah that the solar panel would provide in a day.

$$\frac{9.935Ah}{0.6667A} = 14.903h$$

In Section 5.5.3, we calculated that the worst case theoretical power consumption for each camera node was 1.425 Ah. The final calculation below shows how many hours it would take the solar charger to replenish the Ah used during one cycle using the maximum current that the panel can source.

$$\frac{1.425Ah}{0.6667A} = 2.14h$$

These calculations show that even for the month that produces the least amount of solar energy, it is still possible to recharge the Ah consumed over one day when the camera node is monitoring

the car park for eight hours. We know that this is only a theoretical calculation and does not represent what would happen in actual situations; however, we concluded that the smaller solar panel would work for our application. We chose to use the 6V, 4W solar panel in our design and decided that testing the solar panel would give more accurate results as to whether or not the solar panel would be sufficient. The testing methods and results appear in the Implementation and Results chapter of the report. We chose a charge controller that was compatible with this solar panel; it was the only 6V charge controller sold by Silicon Solar. The controller can handle up to 2A of current. which is less than the solar panel is capable of sourcing. Figure 12 represents the solar panel and charge controller that have been chosen.

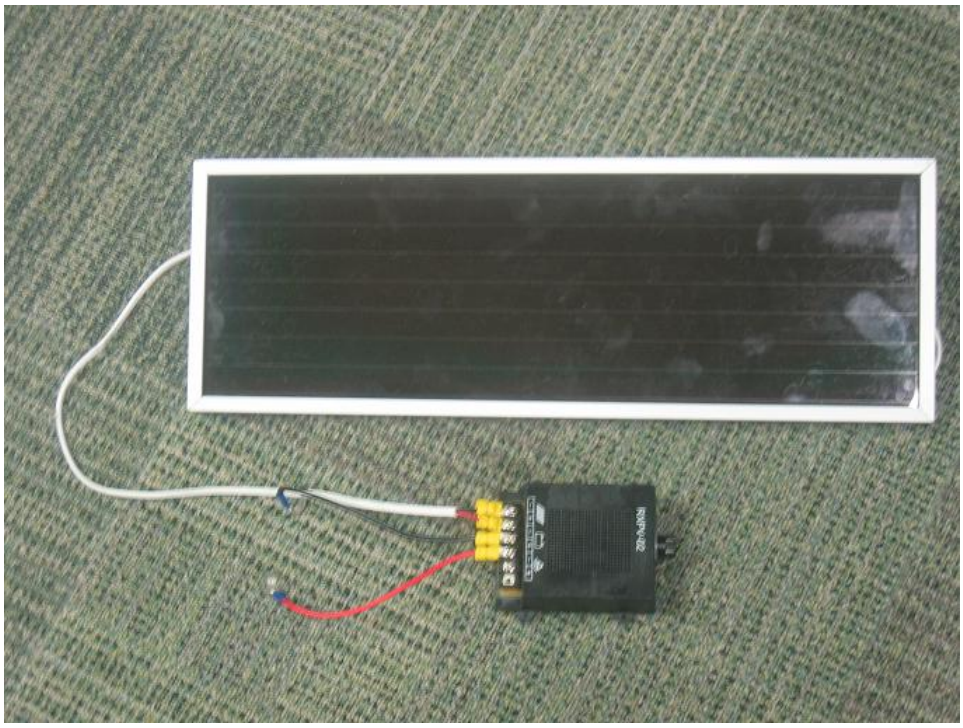


Figure 12: 6V, 4W solar panel.

5.7 Base Station

The functions of the base station included storing images, running a website and notifying the appropriate personnel of activity in the car park. In previous sections, we described in detail the requirements for the base station and possible solutions for implementing those functions. In the Design Approach chapter, we concluded that using a combination of both customized options and external services would suit our need for achieving performance in a relatively short amount of time. This section describes what options were available to fulfill each aspect of the base station, pursuing the respective chosen design paths, and what we decided to use.

5.7.1 Base Station Component Options

As the Enterprise Research Centre provided a desktop PC for the base station and the IT department provided web hosting upon request, we did not consider any more options for these portions of the base station. Both of these options would require little effort or time and provide all necessary functionality.

We chose to use a more customizable design approach for web design. This approach would involve using a scripting language or content management system to set up the website to interface with an end user. While we had experience programming, none of us had experience with any scripting languages. Therefore, we had to consider the additional time commitment for learning the language before beginning to code. We would have to choose a language that was relatively easy to learn, but still powerful enough to support dynamic behavior. Likewise, the content management systems can support web page developers, but some may be difficult to learn how to use.

Content management systems can be divided into two categories: proprietary and open source. The main drawback with proprietary packages was the cost. While these packages had many features and available technical support, the costs for the software for a single installation of Microsoft's SharePoint Design and Expression Web were about €300 and €200, respectively. Another drawback was compatibility; new releases of this software may not be compatible with the version we would use, nor could all platforms use this software. We also found open source packages available to help beginners. Such packages included Joomla, Drupal and Plone. According to an article published by TechSoup, Joomla would be the easiest for the beginner, but would require considerable effort to accomplish dynamic content. Plone would be the most difficult to learn, but has many varied features and a robust reputation. Learning to use Drupal would be not as difficult as Plone or as easy as Joomla and it has features out-of-the-box. [7]

If we were to design the website without any assistance software, we would use a scripting language. Some popular scripting languages were Hypertext Preprocessor (PHP), Perl, Active Server Pages/ Visual Basic Script (ASP/VBScript), and Java Server Pages (JSP). As Microsoft developed the ASP framework, we had some of the same concerns about it as we had with SharePoint Design and Expression Web, in that additional features and support usually come with fees. The other three options were all open source. They each had a fairly good set of features with plenty of available technical support. Notably, according to an article published by GBdirect, learning JSP would require much more time to gain a working understanding of the language and with all the available features. Perl would also require a relatively significant amount of time to learn, but it is most popular among developers of complex and changing websites. PHP lacks the breadth in its libraries that JSP and Perl have, but it would be easier to learn, requiring less time to design a relatively simple website [35]. Though using these languages would be free, we still considered minimizing the amount of required time for developing the website. We found that it would be likely that learning a scripting language may take less time than learning how to manipulate a content management system.

Also, we chose the custom path for sending alert messages, in that we did not believe this proof-of-concept project warranted purchasing an MMS service. The options were sending SMS messages or email messages. SMS messages could alert the appropriate security personnel of new activity and that the pictures could be viewed on the security system's website. In contrast, the email alerts could show the personnel one or more images in the series capturing the new activity. Notably, only personnel with Smartphones could access email from any location, unlike SMS messages which one could receive on all mobile phones. Sending a text message from a computer is much like sending an email; many service providers have email addresses already set up for a computer to communicate with a mobile phone. For example, the format for the email address in the Meteor mobile phone network of Ireland that allowed one to send a text message via email was `10-digit-number@sms.mymeteor.ie`. With completing the description of our base station design options, we describe our design decisions in the next section.

5.7.2 Select of Base Station Components

This section describes the specific design decisions for the base station concerning base station hardware, web hosting, website design, and sending alert messages. While making these decisions, we looked for an option that would fulfill our functionality requirements and would not take a considerable amount of time.

As this is a proof-of-concept project, we did not feel building a custom base station was necessary, so we chose an existing computer for the base station. In addition, we did not require much storage or much traffic support from a web hosting service. As the Enterprise Research Centre provided a computer and the IT department provided web hosting, our decisions here were simple. The group chose the resources made available by the University.

As we pursued a customized option for web design, we chose to use PHP as our scripting language. While it would still require us to take the time to learn this scripting language, it was free, and also flexible enough for us to accomplish our functions for the website.

Regarding alert messaging, we chose to write scripts for both emails and text messages. Though making pictures immediately available to personnel would be effective for security, we did not feel it was necessary to demonstrate this service for a project at this stage of development. Individually, the SMS messages could notify personnel at any location, and the emails could provide visual information of the activity. With our design explained, we proceeded to order the necessary parts and began learning about the implementation of this design. We described in the next sections how we went implemented the device and the resulting behavior of the system with our these design choices.

6 Implementation and Results

With our design decisions set, we ordered our components and modules. See Appendix A for a complete parts list. Upon arrival of the parts, we proceeded to test our components and modules to ensure they would perform sufficiently for this surveillance system project. Once we tested their performance individually, we then built the prototype in blocks. For each block, we tested the interface and overall functionality. Putting all blocks together, the team achieved system integration.

6.1 System Overview

This section briefly presents the audience with an overview of our group's design. The proof-of-concept camera node consists of a motion sensor, a camera module, a MSP430 development board, a SD memory card, and a power source. The power source consists of a solar panel, a charge controller, and a lead acid battery. Figure 13 depicts the top-level block diagram of this system.

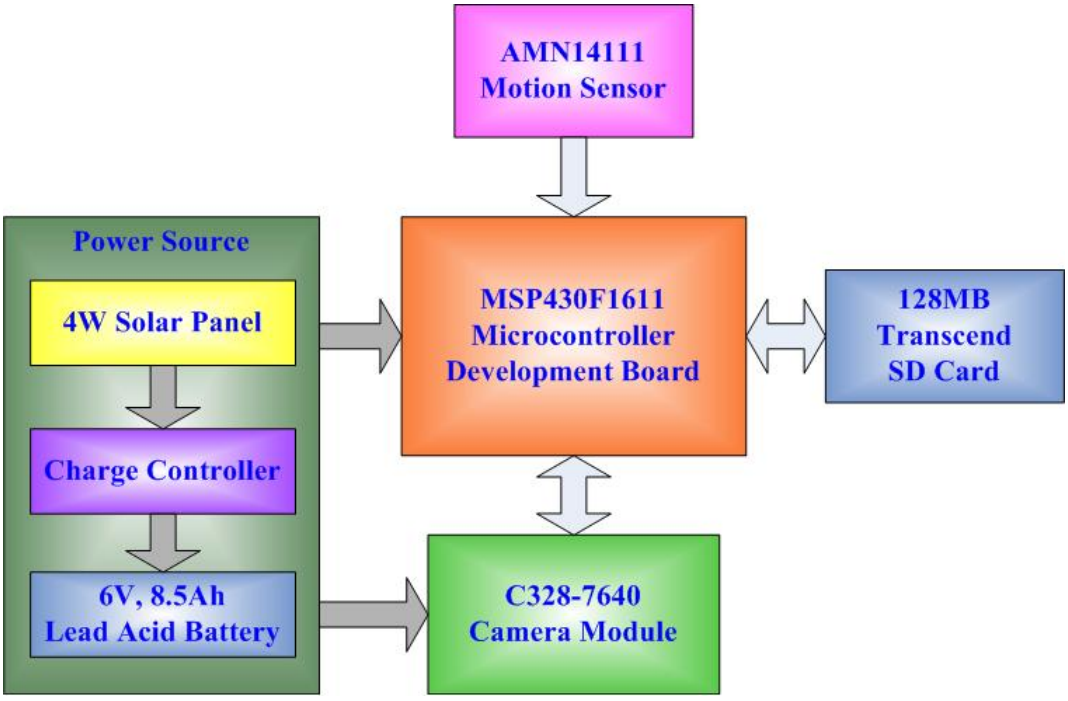


Figure 13: System overview.

As seen in Figure 13, the motion sensor would trigger the MSP430 microcontroller which, in turn, requests the camera module to take a specified number of pictures. The MSP430 would then transfer the taken images to the SD card. The system entered the sleep mode when no activity occurs. Finally, the power source powered the camera module and the development board.

6.2 Components Testing

In this section, we describe our testing results with each component or module of the camera node. Then, we compared these results to the theoretical performance described in the documentation to conclude if the performance is acceptable compared to the theoretical performance.

6.2.1 Motion Sensor Testing

To verify that the AMN14111 PIR sensor from Panasonic functioned, we documented the performance of the PIR sensor at several supply voltages. The datasheet stated that the sensor would indicate motion by its output transitioning from 0V to the supply voltage [28]. The acceptable range of supply voltages are 3V to 6V. At 3V, 6V and also 3.3V (the voltage supplied by the MSP430 development board), we observed proper functionality as a hand was waived in front of the sensor. After motion ceased, it took about one second for the voltage to return to 0V. In addition, we recorded the current draw of the PIR sensor at each voltage level. The results appear in Table 18. As we desired to be energy efficient with this prototype, the low current draw was acceptable for this design.

Table 18: Current draw of PIR sensor at various voltages.

Supply Voltage (V)	Current Draw (μA)
3.0	99
3.3	107
6.0	170

By using PIR sensors, we hoped to have blanket coverage of the Foundation Car Park for sensing motion. In the documentation, the range of detection for the PIR should be 10m with a horizontal range of 110° and a vertical range of 93°. To determine how effective this range would be, we surveyed the car park. We determined what was a basic unit of the car park, such that the lamp post configuration was repeated throughout the car park. We took some measurements of distances within this basic unit so as to estimate the effective coverage of the PIR sensor. Figure 14 shows this basic unit of the car park from above with the theoretical coverage of the horizontal range of the PIR sensors. The black circles represent lamp posts. The ranges of the PIR sensors have been drawn here, shown in yellow, green and blue for clarity, with the intention of covering the most area. The red highlight areas that are not covered by the PIR sensors.

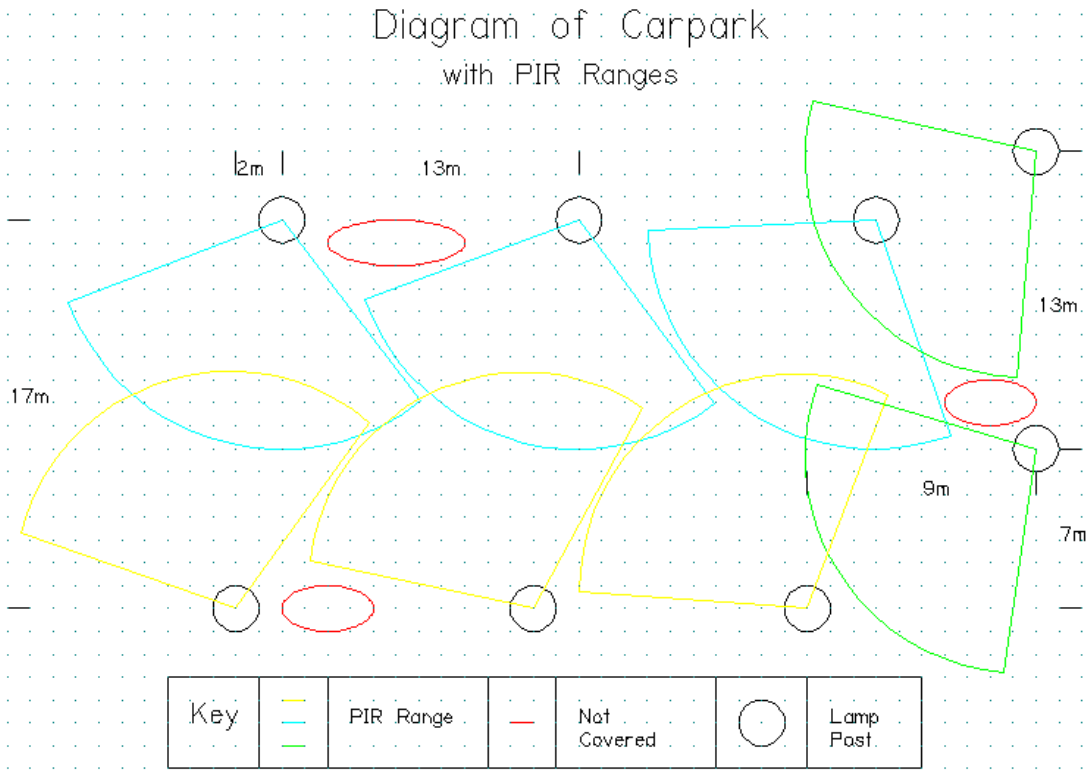


Figure 14: Overhead view of car park with PIR sensor range.

As we hoped to have blanket coverage, this theoretical situation was acceptable. The areas not covered by sensors were relatively small and isolated, such that any suspicious individuals could not avoid the sensors while moving around the car park by staying within these areas. In addition, we charted the theoretical coverage of the vertical range of the PIR sensors. As one can see in Figure 15, the range should cover all area between lamp posts in a car park lane.

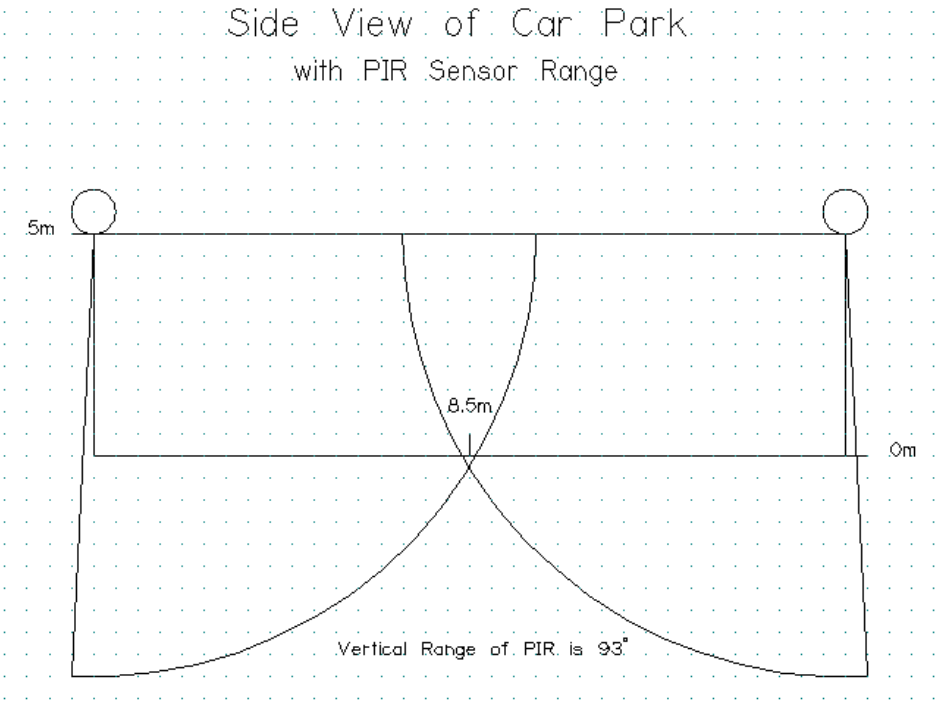


Figure 15: Side view of car park with PIR sensor range.

To document the actual performance of the PIR sensor, we tested the sensor in a situation that was similar to the proposed installation. We mounted the sensor about 4.4m above an open area angled 45° down. As the lamp posts were about 4m tall, this height represented slightly more than the worst case for the distance from activity in the car park. The motion trigger in this test was a person walking. We tested its sensitivity on both the left and right sides; we marked the distances at which it first detected motion as a person walked toward the middle of the testing area. We repeated this process three times at each successive meter until the sensor no longer registered motion. Tables 19 represents the data that we collected. We were not able to collect data for two points on the left side of the PIR sensor because there was an obstruction.

Table 19: Data from PIR sensor range test.

Distance from PIR sensor (cm)	Avg. Distance Left of Center (cm)	Avg. Distance Right of Center (cm)	Range on Left center	Range on Right	Horizontal Viewing Range
100	120.7	351.3	50°	74°	124°
200	224.7	215.3	48°	47°	95°
300	144.3	520.0	26°	60°	86°
400	484.3	629.0	50°	58°	108°
500	659.0	758.3	53°	57°	109°
600	332.3	420.7	29°	35°	64°
700	361.0	578.7	27°	40°	67°
800	432.3	626.0	28°	38°	66°
900		727.3		39°	39°
1000		665.7		34°	34°
1100	531.3	722.0	26°	33°	59°

We graphed the data to show the actual results versus the range suggested in the documentation. Figure 16 plots the data we collected against the expected results. In addition, Figure 17 plots the left, right, and total viewing ranges and shows the general trend as one moves further from the PIR sensor.

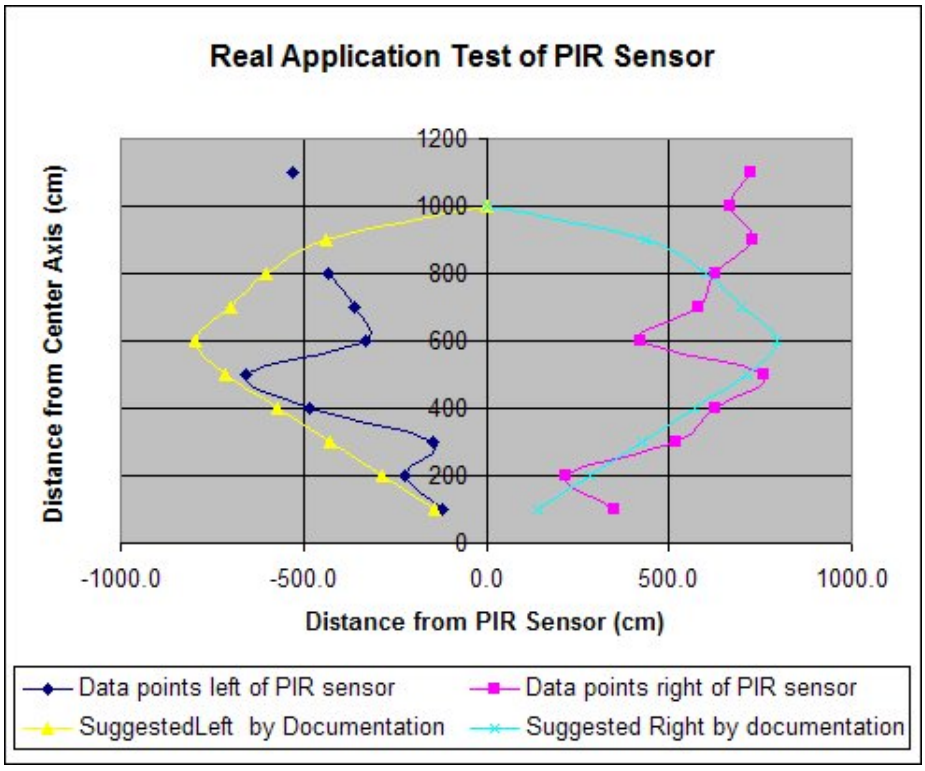


Figure 16: Chart of PIR sensor testing results.

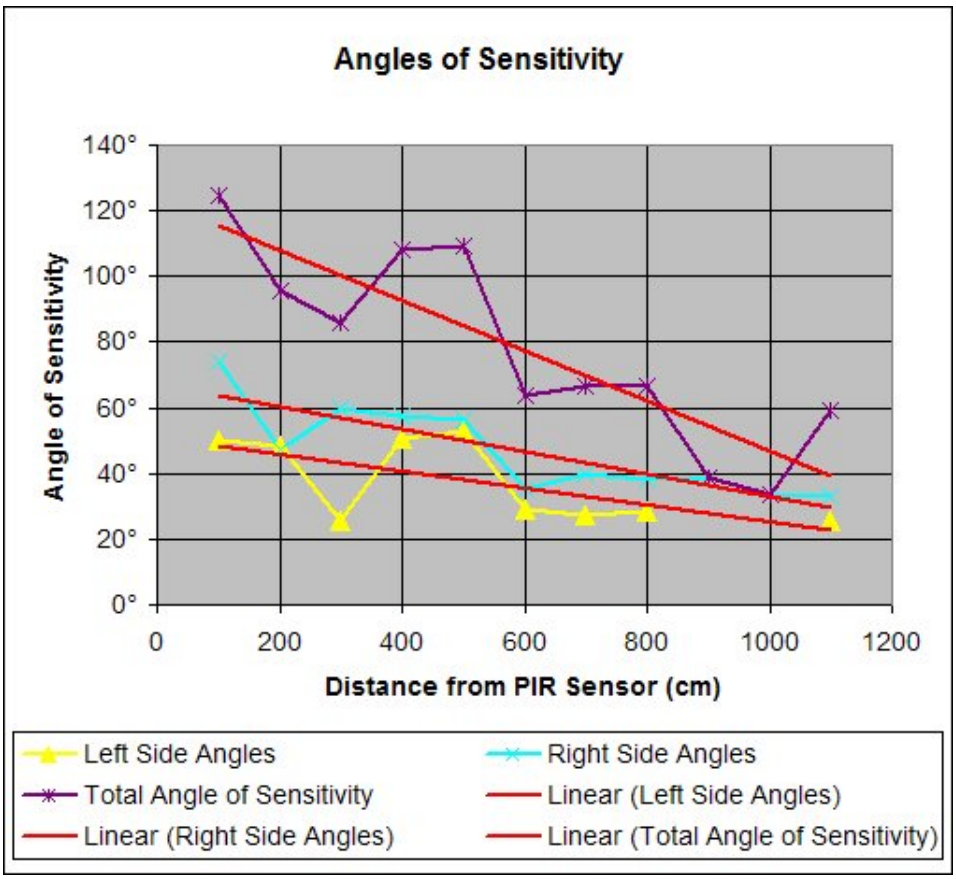


Figure 17: Plot of PIR sensor ranges.

In addition, we attempted to document the blind spot behavior of the PIR sensor. If the motion within the range of the PIR sensor is directly toward the sensor, it may not detect the motion immediately as not enough of its pixels change to indicate motion. Though this behavior was not consistent, we plotted a few points against the expected range to show this limitation of the sensor. Figure 18 shows this data collected about the blind spots.

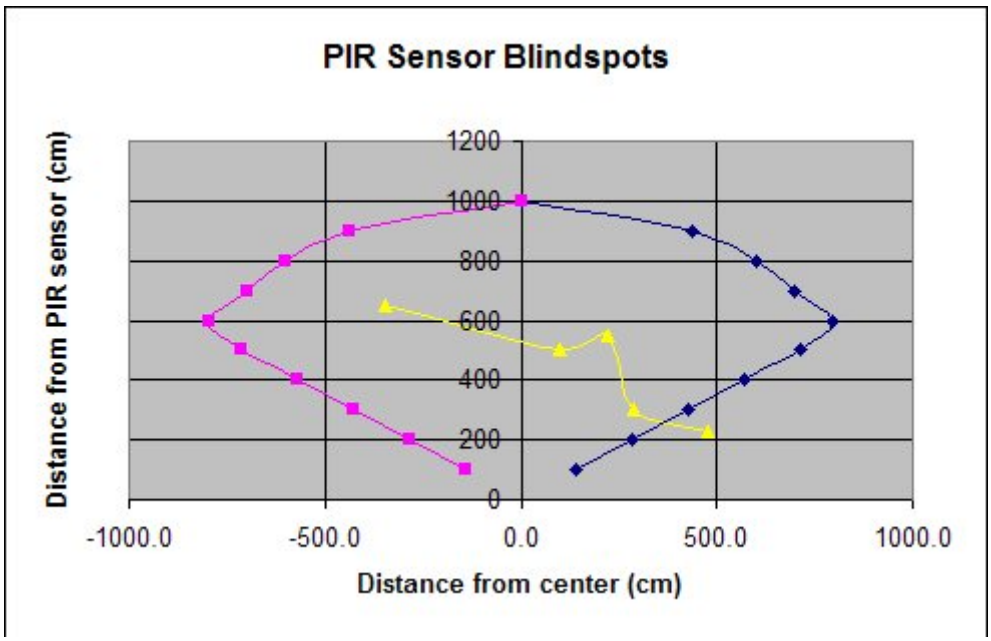


Figure 18: Plot of PIR sensor vertical range measurements.

In conclusion, the area in which the PIR sensor detected motion was relatively close to what the documentation indicated, but in a different shape. Instead of a consistent detection range between 110° and 93° , the sensitivity ranged from 124° and 34° . This performance was acceptable for proving the concept of this surveillance system project.

6.2.2 Camera Module Testing

This section describes the procedures for examining the camera module C328-7640. We tested the module by using the C328-EV232 evaluation board which came with an RS-232 serial cable, a 5.0V DC adapter, and a CD that contained datasheets and a program for testing the camera module. Figure 19 shows the evaluation board.

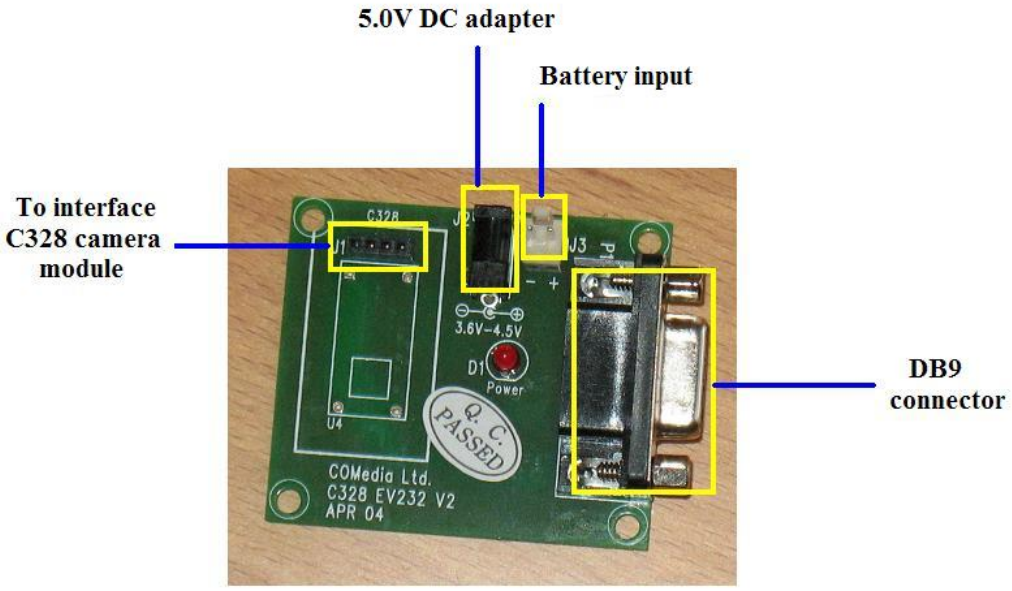


Figure 19: C328-EV232 evaluation board.

We used an RS-232 serial cable to connect the board to a desktop computer. Figure 20 illustrates this setup.

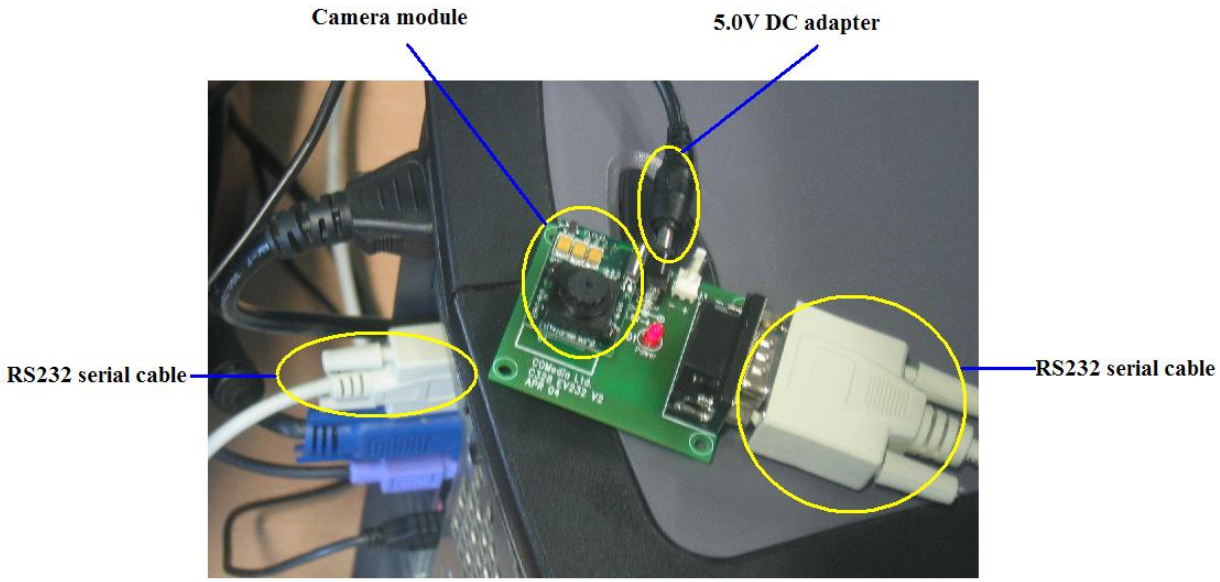


Figure 20: Testing camera module.

We made use of the provided software on the CD to test the camera module. The user interface of the program is shown in Figure 21.

Figure 21: Camera evaluation program.

The program continually requested pictures from the camera module. It showed a preview of each picture on the left side. The user could obtain a JPEG snapshot by clicking the "Go!" button. The program then obtained a snapshot, saved the file, and displayed the image on the right side. A user also had the option of selecting the snapshot resolution. Several pictures with various resolutions captured by this program can be found in Appendix D of this report. Not only did we use the evaluation board for testing the camera module, but we also utilized it for interfacing with the MSP430 development board. We described this interface in Section 6.3.2.

6.2.3 Development Board Testing

Upon the arrival of the MSP430 development board, we conducted four tests in order to verify the normal operation of the board. In this section, we describe those tests and their results. To help the audience understand the purpose of the tests, we briefly explain each test below.

Power supply and current draw test: This test was to ensure that the board would turn on with either an AC adapter or a DC source. Additionally, we measured the current draw of the board

Timing test: The MSP430 microcontroller has a built-in timer. We used this feature later to program a sleep mode based on an elapsed period of inactivity.

Interrupt capability test: We used two interrupts on the MSP430, therefore we first tested the response capability of the microcontroller interrupts from the peripherals.

UART test: We used the UART interface to communicate between the camera module and the MSP430, so we first verified its functionality.

In the power supply and current draw test, we used an AC adapter, a DC power supply, and a digital multimeter. The documentation for the development board stated that the appropriate AC adapter for the board had an output voltage of 4.5V to 6.0V DC. As the operator will be using batteries to power the nodes, we also tested the board without the AC adapter and just a DC source. We used an AC adapter with 5.0V DC output to power up the board. Upon connecting the board to the AC adapter, both the power LED and status LED lit up. We then switched the AC adapter to a DC power supply with 5.0V output voltage. In this case, both LEDs also turned on. Both sources had the same behavior on the board thus far.

We measured the current consumption of the board. It should be noted that the power LED was always active as long as there was power on the board, though the status LED could be disabled by pressing and holding the on-board user button. Hence, we wanted to obtain the current consumption of the board in two cases: a case with active status LED and a case with inactive status LED. Table 20 shows these measurements.

Table 20: Current consumption of the MSP430F1611 development board.

	Current consumption (mA)	
Supply voltage (V)	With active status LED	With inactive status LED
4.50	7.710	6.636
4.75	8.543	7.034
5.00	9.858	7.777
5.25	10.012	7.356
5.50	11.336	8.134
5.75	12.703	8.724
6.00	12.851	8.725
6.25	12.853	8.725

Table 20 indicates that the status LED consumed roughly between 1mA and 4 mA within the board's operating voltage range. This amount of current was a considerable portion of the board's total current draw. In conclusion, the status LED is useful for testing and debugging purposes; however, it should be disabled in actual operation to achieve greater power efficiency.

For the timing test, we examined the built-in timers of the MSP430 microcontroller. The documentation stated that there are two timers that are implemented in this chip: timer A and timer B. Timer B is similar to timer A with a few more features. These timers function as counters which have a length of up to 16 bits. In other words, they can count up to a maximum of (2^{16} - 1) or 65535. In order to see how the timer B worked, we decided to toggle the status LED at a frequency of 1Hz, i.e. the LED would be blinking once every second. Figure 22 illustrates the program flow.

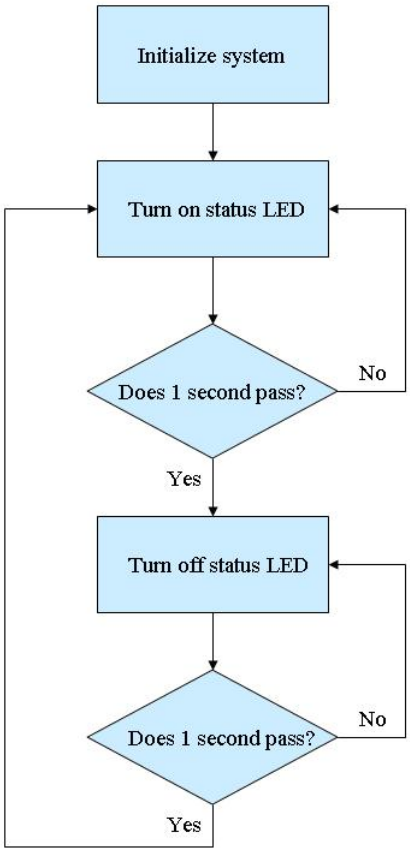


Figure 22: Timing test flow.

In Figure 22, the first block indicates that the system must be initialized prior to running the test. The maximum size of the timer, 65535, limits our range of frequencies. Therefore, we decided to use 50kHz. After the timer counted to 50,000, 0.1 seconds had passed and a variable named timeflag would be incremented. After the timeflag variable increased to 10, one second had passed and the status LED would toggle. As a result, the LED would blink at a frequency of 1 Hz.

In the interrupt capability test, we examined the interrupt capability of the microcontroller. Before presenting the details of this test, we define an interrupt for the audience; basically, an interrupt is an external signal that requests the microcontroller's attention. We used an interrupt to control the sleep mode, used to conserve energy. As discussed in previous chapters, if the camera node system is in sleep mode, it can only be awakened by the motion sensor. When the PIR sensor detects motion, its output voltage would switch from a low logic level to a high logic level. The low-to-high transition would trigger the microcontroller. The microcontroller would respond to the interrupt and exit the sleep mode so that the system could start taking, processing, and storing photos. Since we were only testing the board at this point, we created a simple circuit, which emulated the motion sensor, to manually generate interrupts. Figure 23 shows this circuit.

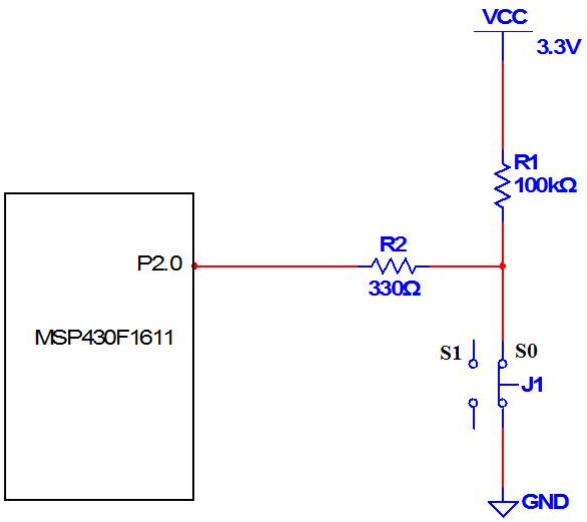


Figure 23: Circuit that emulates the motion sensor.

This simple circuit that simulated the motion sensor consisted of R1 and R2 resistors, and a switch J1. The switch J1 could be at either position S0 or S1. Initially, J1 stayed at S0; hence, pin P2.0 (pin 0 on port 2) of the microcontroller was connected to ground through R2. As soon as J1 switched its position to S1, the voltage at pin P2.0 would be pulled up to V_{cc}. As a result, we obtained a low-to-high transition by changing the position of J1 from S0 to S1.

Generally, the microcontroller would enter sleep mode after being powered up for ten seconds. That would also turn off the status LED. If a low-to-high transition generated by the switch J1 occurred at pin P2.0, the system would be activated and the LED would be lit up. Then, the microcontroller would go back to the sleep mode after 3 seconds. And again, it would be awakened if it received an interrupt signal. Figure 24 illustrates the flow of the test.

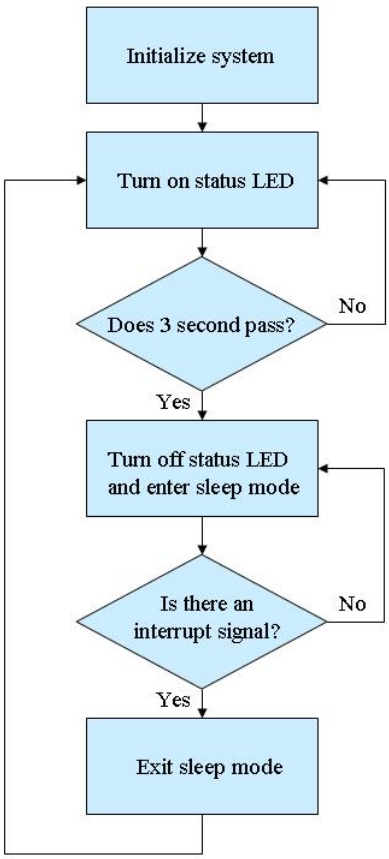


Figure 24: Interrupt capability test.

The system had to be initialized prior to performing the test. We configured Timer B to be the same as the timing test and we configured the pin P2.0 of the microcontroller to detect an interrupt signal. The microcontroller responded to the interrupt in this test.

In the final UART test, we examined the UART transmission of the MSP430. It should be noted that the chip has two UART ports: UART0 and UART1. We initialized the UART0 in this test as it was pre-built with a 9-pin D-sub connector. This connector enabled us to make a direct connection between the development board and a desktop computer. In order to use UART0, one would have to configure several parameters. Table 21 presents the value of those parameters.

Table 21: UART0 configuration.

Parameter	Value
Baud rate	38400 bps
Data	8 bits
Parity	No
Stop bit	1

The goal of this test was to have the MSP430 echo the characters received from the computer. To perform this test, we utilized software named COM Port Toolkit, used to monitor, log, and capture data transmission on serial port. Figure 25 shows the results of the data transmission between the microcontroller and the computer.

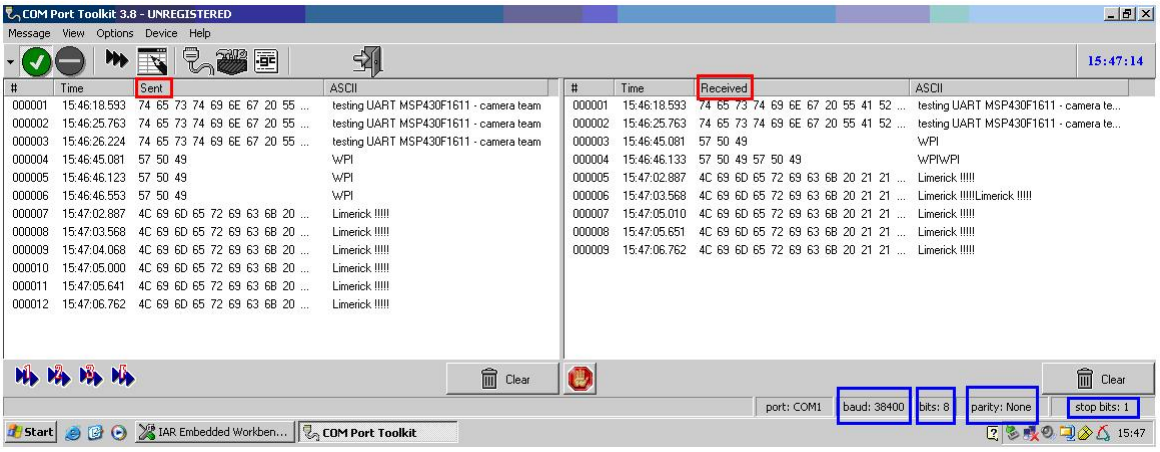


Figure 25: Results of the UART test.

The left hand side of the program's window displays the data that the computer sent to the microcontroller. The echo appears on the right hand side. The information on the bottom right corner of Figure 25 confirms the UART0 parameter values.

In summary, we completed all four tests successfully on the development board. Through these tests, we gained a basic understanding of the microcontroller's functionality. Furthermore, we reused the configurations of each test in the actual software program of the project.

6.2.4 Memory Testing

In this section, we briefly describe the testing of the memory card. To verify the functionality and the size of the SD memory card, we inserted it into a card reader, which we connected to a desktop computer via a USB port. The original capacity of the SD card was 121 MB, which was less than the capacity - 128 MB - specified in the card's datasheet. However, it is common that the actual capacity of a storage device is often smaller than its specification. To verify reading and writing, we copied an image from the computer's hard disk and pasted it onto the card.

The size of the image was 32KB. We were able to view the image from the SD card. With the card reader, the SD card performed basic functions.

6.2.5 Power Source Testing

We describe the method for testing the power components used to supply each camera node in this section. As previously stated, the power source consists of three parts: solar charger, charger controller and lead acid battery. Figure 26 depicts the configuration of these devices.

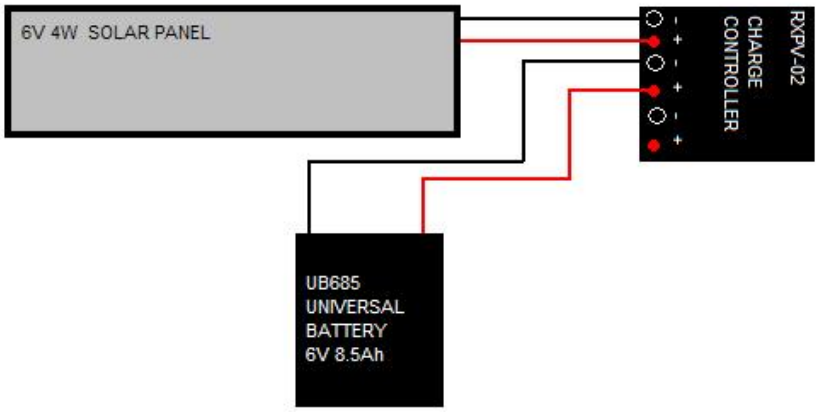


Figure 26: Power supply component configuration.

In Figure 26, one can see that one set of terminals connects to the battery, one connects to the solar charger, and one set remains unconnected. These terminals on the charge controller are meant for the load of the system. We did not use these terminals; upon testing the controller, we found that these terminals did not supply 6V while the battery was charging. Since part of the camera node would always need power, the node should be connected directly to the battery. This configuration does not prevent the charge controller from performing its main functions: protecting the battery from overcharge and preventing the battery from discharging through the solar panel. The development board has built-in voltage regulators that condition the battery voltage to safely power the rest of the system.

We tested the battery to determine its usable capacity. To drain it, we connected it in series with a resistive load of 6.43Ω. We used a multimeter to measure the current drawn from the battery and the voltage of the battery at half hour intervals until the current dropped to 82mA and the voltage dropped to 2.44V; conditions that would be insufficient to power the camera node. We derived from the collected data an estimate of the Ah capacity of the battery. Simultaneously monitoring the current and voltage also allowed us to determine the usable Ah of the battery. The mesh network team needed at least 5.5V to power their transceiver module. The point at which the battery voltage dropped below this value was the point at which there would be no more usable Ah. Table 22 displays the data that we collected.

Table 22: Voltage and current vs. time while draining the battery.

Duration (min)	**Current (mA)**	**Voltage (V)**
0	918	6.387
30	892	6.271
60	928	6.212
90	897	6.143
120	898	6.103
150	898	6.065
180	915	5.980
210	903	6.076
240	868	6.009
270	865	5.955
300	860	5.896
330	864	5.840
360	875	5.794
390	845	5.691
420	842	5.610
450	760	5.261
480	617	4.401
510	438	3.320
540	374	2.876
570	304	2.665
600	280	2.556
630	212	2.543
660	125	2.593
690	82	2.440

To calculate the total Ah with the information collected, we first summed the currents measured. Since this value was the mA for every half hour, we divided it by 2; this calculation gave us the mAh. We then divided this number by 1000 to obtain Ah.

$$I_{total} = 16460mA$$

$$TotalmAh = 8230$$

$$TotalAh = 8.23$$

To determine the usable Ah from the battery, we summed the currents until the voltage reached about 5.5V; our closest measurement to 5.5V was 5.61V. Again, we divided this number by 2 and then by 1000 to achieve a value in Ah.

$$I = 13268mA$$

$$mAh = 6634$$

$$Ah = 6.63$$

Figure 27 depicts the data in the Table 22. The two curves follow the same pattern and it is noticeable from the graph that both curves break at about the same point; this point also represents our cutoff for usable capacity and it is marked with a line.

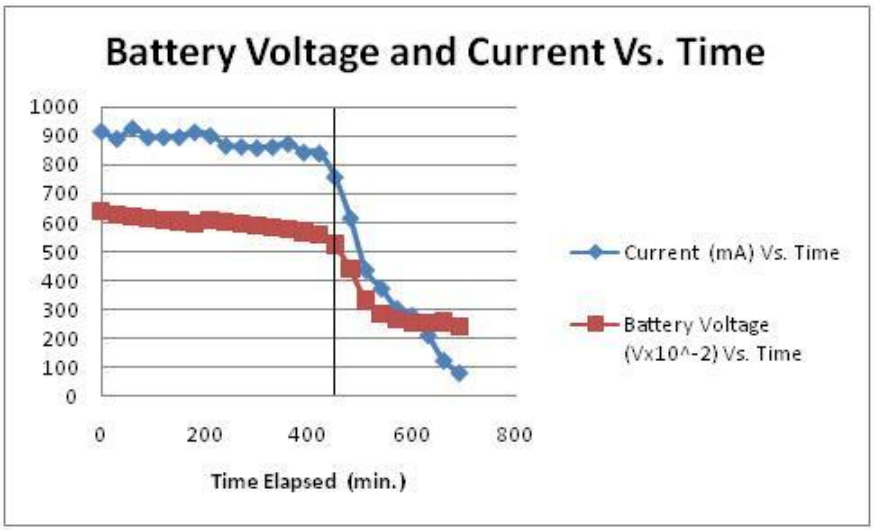


Figure 27: Battery voltage and current vs. time.

We estimated how many days the battery could power the camera node without any recharge with this usable Ah value and with the power consumption estimations from Section 5.5.3. When the system is active 50% of the time and in sleep mode 50% of the time, we found that the system could last approximately 7.9 operating days.

$$\frac{6634mAh}{\frac{840mAh}{50/50day}} = 7.9days$$

When the system operates under typical conditions of 99% in sleep mode and 1% active we found that the camera nodes should last about 17.46 days without recharging the battery.

$$\frac{6634mAh}{\frac{840mAh}{99/1day}} = 17.46days$$

We also used the multimeter to gather data about the effectiveness of the solar panel. The situation we tested was a day in which the conditions ranged from partly sunny to overcast. In this case, we monitored the current that the solar panel was able to source to the battery every minute for one hour. This procedure gave us data on how much the current can change in one hour, in addition to the Ah for an hour with these weather conditions. The data is represented in Table 32 in Appendix C and plotted in Figure 28.

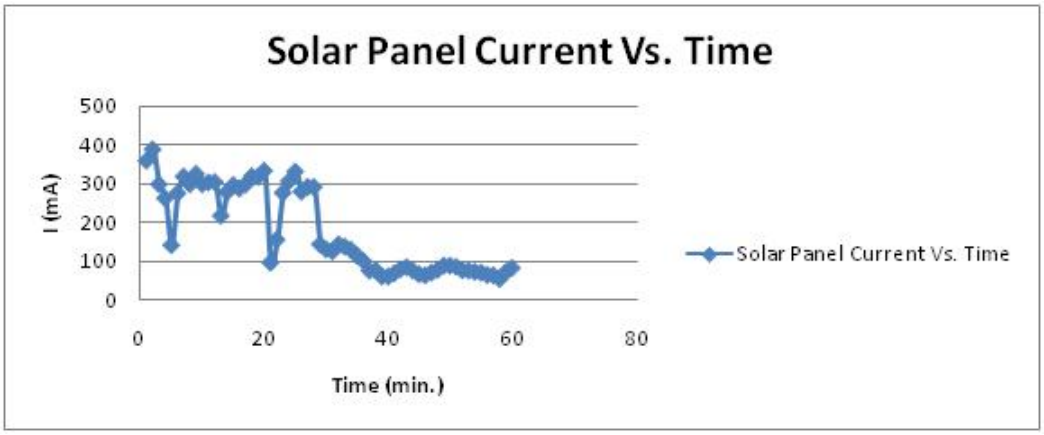


Figure 28: Current sourced by solar panel vs. time.

In this case, the solar panel was able to provide 180.35mAh. Since this only accounted for one hour, however, we examined a second case: a typical day in September. Over the course of twelve hours of sunshine, we measured the solar panels performance about every half hour. The weather conditions changed from clear and sunny to cloudy. Figure 29, with Table 31 located in Appendix C, display the data collected.

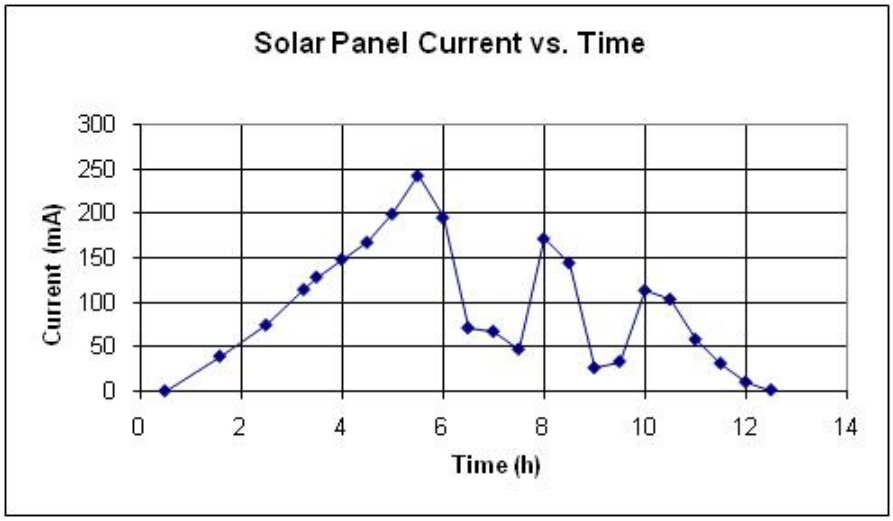


Figure 29: Solar charging data over 12 hours.

The total mAh that we calculated from the data collected amounted to 1118.42mAh; we estimated the draw of the camera node during an eight hour cycle at 50% active, 50% in sleep mode. This value was 840mAh. The information we collected shows that the solar panel can overcompensate for the power used in that operating day. We anticipate that in actuality, the camera node would be active 1% of the time and be in sleep mode 99% of the time, which would consume about 380 mAh.

To determine the performance of the solar charger in the month with the least amount of solar radiation, we referred back to Table 17. The solar radiation for the month of September is $2.61\frac{kWh}{\frac{m^2}{day}}$. December, the month that has the least amount of solar radiation, produces on average $0.46\frac{kWh}{\frac{m^2}{day}}$. We used the collected data for September and the ratio of September's and December's insolation values to estimate the Ah that the solar charger would likely produce in December.

$$\frac{2.61\frac{kWh}{\frac{m^2}{day}}}{1118.42mAh} = \frac{0.46.61\frac{kWh}{\frac{m^2}{day}}}{x}$$

x = mAh for the month of December = 197.12mAh

Unfortunately, this value is fewer mAh than even the typical cycle will consume; the typical cycle consumes about 183mAh more. Referring back to the total amount of usable mAh for the battery, it had about 6634mAh. At a loss of 183mAh each day, the battery should be able to power the camera node for about 36 days. In addition, we determined what other months would not have enough sun to sustain the power supply. Table 23 shows our calculations for the six least sunny months.

Table 23: Calculations for months with low solar insolation.

Month	Insolation $\frac{kWh}{\frac{m^2}{day}}$	Estimated Available mAh	Deficit (mAh)
October	1.37	587	+207
November	0.71	304	−76
December	0.46	197	−183
January	0.64	274	−106
February	1.10	471	+91
March	2.03	869	+489

To summarize Table 23, a user would have to charge the battery with utility power about once a month during November, December, and January.

6.3 System Integration

In this section, we describe our interfaces within the system. In addition, we describe how we tested these interfaces and our results.

6.3.1 Interface between Motion Sensor and Development Board

This section presents the interface between the PIR sensor and the development board. The PIR sensor has three pins: ground, V_{DD}, and output. As pictured in the schematic in Figure 30,

the pins 1 and 2 connect to V_{DD} and ground respectively. Though not depicted in the schematic, the V_{DD} and ground are pins on the development board. The output pin on the PIR sensor connects to P2.0 on the development board.

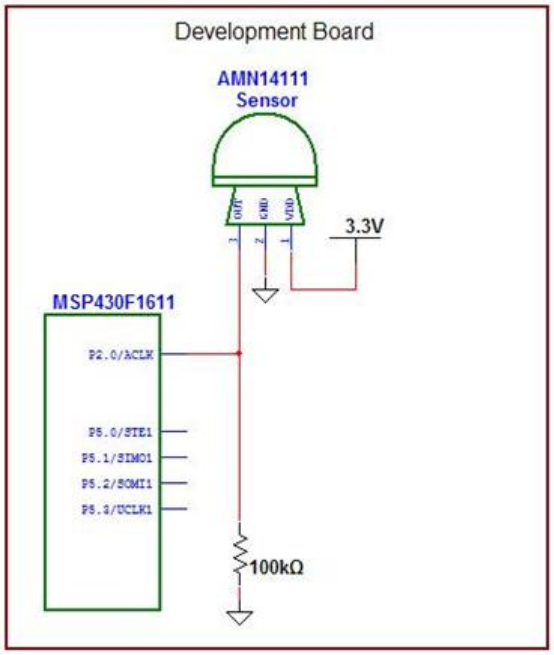


Figure 30: Schematic of PIR sensor and development board.

Upon sensing motion, the PIR sensor would be triggered and the voltage on P2.0 would be pulled high to V_{DD}. This is the switch from logical low to high that we simulated when testing the interrupt on the microcontroller. The microcontroller would process this transition from low to high as an interrupt and wake up the system from sleep mode. The flow chart in Figure 31 represents the motion sensors role in the overall system. When the system initializes, it configures pin P2.0 for sensing an interrupt. The system would then run in sleep mode. If the motion sensor is not triggered, the system would remain in sleep mode. When the sensor is set off, the MSP430 would read this as an interrupt and activate the system. Once the system executed its necessary activities, it would return to sleep mode.

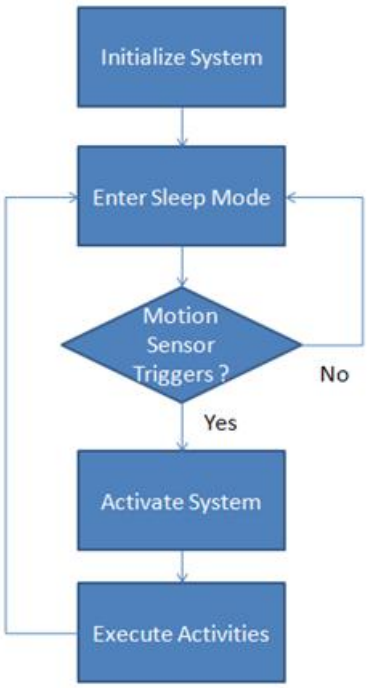


Figure 31: Flow chart of behavior between PIR and development board.

According to the datasheet, the sensor's output is left unconnected to ground when there is no detection. If the previous sensor's output state is in logic high while the current state is in logic low, the wire, which is connected to the output pin, might hold charge due to parasitic capacitance between the wire and other medium. Thus, we placed a pull-down resistor between the output and the ground pin of the sensor in order to discharge quickly any remaining charges on the wire. By doing this, we could be able to obtain a smooth, quick low-to-high transition at the interrupt pin. We decided to use a resistance that was equal to or larger than $\mathrm{V}_{DD}/\mathrm{I}_{outmax}$ of the PIR sensor.

$$\frac{3.3V}{100\mu A} = 33k\Omega$$

We chose to use a 50 kΩ pull-down resistor. Figure 32 depicts the connection between the motion sensor and the development board. The red wire connects to V_{DD}, the yellow wire to ground, and the orange wire to P2.0.

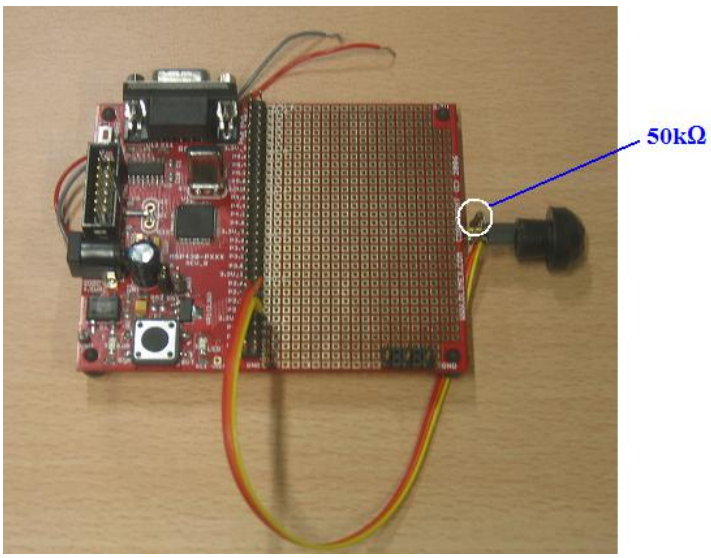


Figure 32: PIR sensor and development board.

6.3.2 Interface between Camera Module and Development Board

This section presents the audience with the details of the integration between the camera module and the development board. It also discusses the procedures we developed to test this integration.

As mentioned previously in Section refD-sub, the C328-EV232 evaluation board had a pre-built 9-pin D-sub (DB9) connector. This simplified the connection between the C328-EV232 board and the development board. Figure 33 and Figure 34 present the schematic of this interface, as well as the actual pieces of hardware.

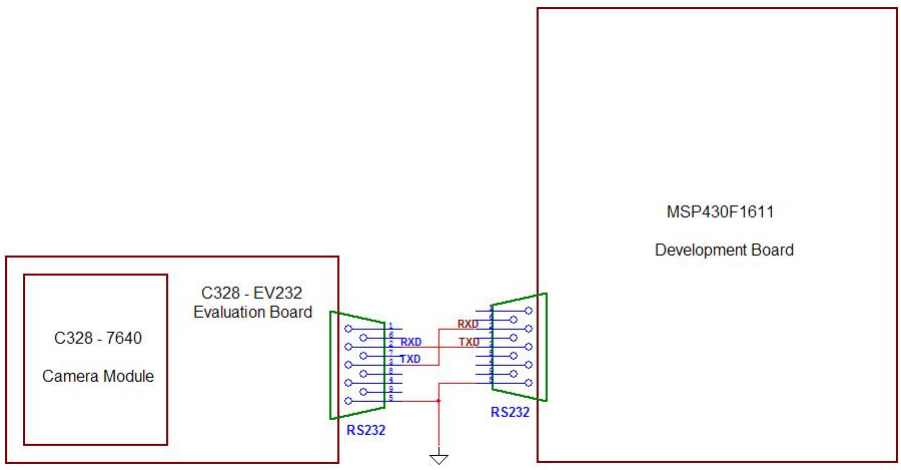


Figure 33: Schematic of the interface between camera board and development board.

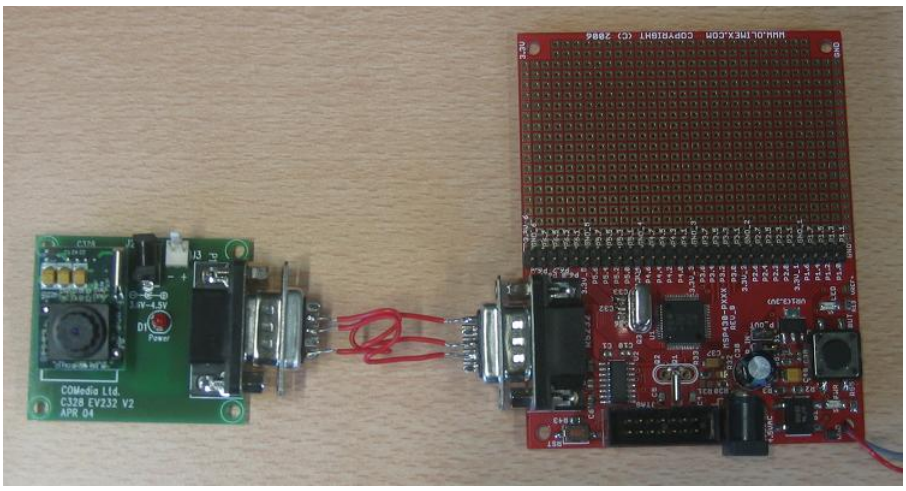


Figure 34: Camera board and development board.

Our overall goal was to have our code running on the MSP430 control the snapshot function of the camera. Prior to writing the code, we reviewed the user manual to gain an understanding of the C328-7640 camera module. This module has a set of user-friendly commands which one would use to interface with an external host, i.e. the MSP430 microcontroller. The host would send the commands to the module in order to control its operation. Each command consists of six continuous bytes. Figure 35 is an example of a synchronize (SYNC) command.

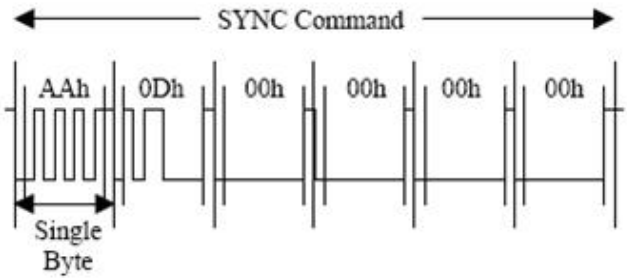


Figure 35: SYNC. http://www.electronics123.net/amazon/datasheet/C328-7640UM_V3.PDF

The first byte of all the commands is defined to be the same: 0xAAh. The second byte is called the ID number of the command. The remaining bytes represent parameters; a user could configure such parameters as color type, resolution, file format, baud rate, and package size. Table 24 shows the commands that we used in this design.

Table 24: Camera module commands.

Command Name	Command Content	Comment
SYNC	AA 0D 00 00 00 00	
ACK	AA 0E xx 00 yy yy	xx: command ID ; yy yy: package ID
Initial	AA 01 00 07 00 07	JPEG format, 640x480 resolution
Set Package Size	AA 06 08 00 02 00	Each data package contains 512 bytes
Snapshot	AA 05 00 00 00 00	
Get Picture	AA 04 01 00 00 00	
Reset	AA 08 00 00 00 FF	
Power Off	AA 09 00 00 00 00	
Data	AA 0A 01 zz zz zz	zz zz zz: image size

Either the host or the camera could establish a connection by issuing the SYNC command. The ACK command indicates the successful receipt a valid command. The microcontroller can also issue this ACK command to request an image data package with the desired package ID from the camera. One would configure the camera with the Initial and Set Package Size commands. The host could trigger the camera to take pictures by sending the Snapshot command. To obtain image data, the microcontroller would need to transmit the Get Picture command, and the camera would send back the Data command to confirm the image size. If the camera fails to operate, the host could restart the camera by issuing the Reset command. Finally, one would use the Power Off command to deactivate the module.

We created a flowchart to define the procedure for taking a picture. Prior to taking a snapshot, the microcontroller would continuously send the SYNC command a maximum of 60 times until it received an ACK and a SYNC command from the camera. The host then would have to reply with an ACK to inform the camera that the connection has been established. After that, other commands could be sent to configure the camera and request image data. Figure 36 illustrates this process.

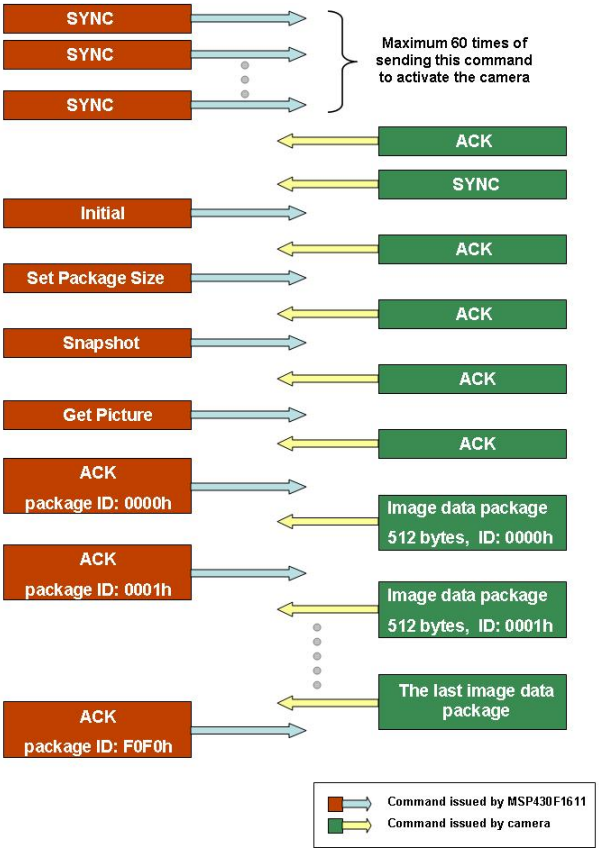


Figure 36: Sequence of taking a picture.

In this test, we decided to make use of the remaining UART1 port. The purpose of the test was to verify that the MSP430 could control the output of image data from the camera module. It accomplished this by sending the data to the computer via UART1 of the microcontroller. Figure 37 illustrates this idea.

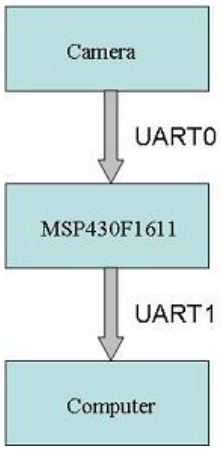


Figure 37: Flow chart of testing camera module and development board.

To make a connection between the microcontroller and the computer, we utilized another C328-EV232 evaluation board since it had a built-in DB9 connector. Figure 38 shows the connections in this testing.

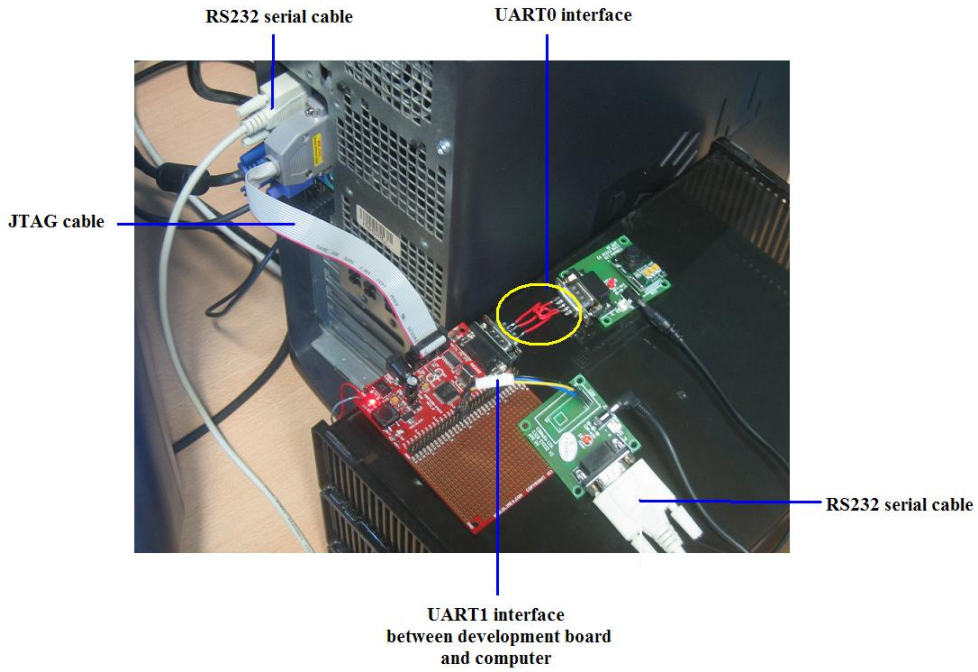


Figure 38: Configuration for camera module and development board test.

The setup in Figure 38 allowed us to test our code, which requested a picture from the camera, and allowed us to view the image data on the computer's screen with the COM Port Toolkit program. Figure 39 depicts this result.

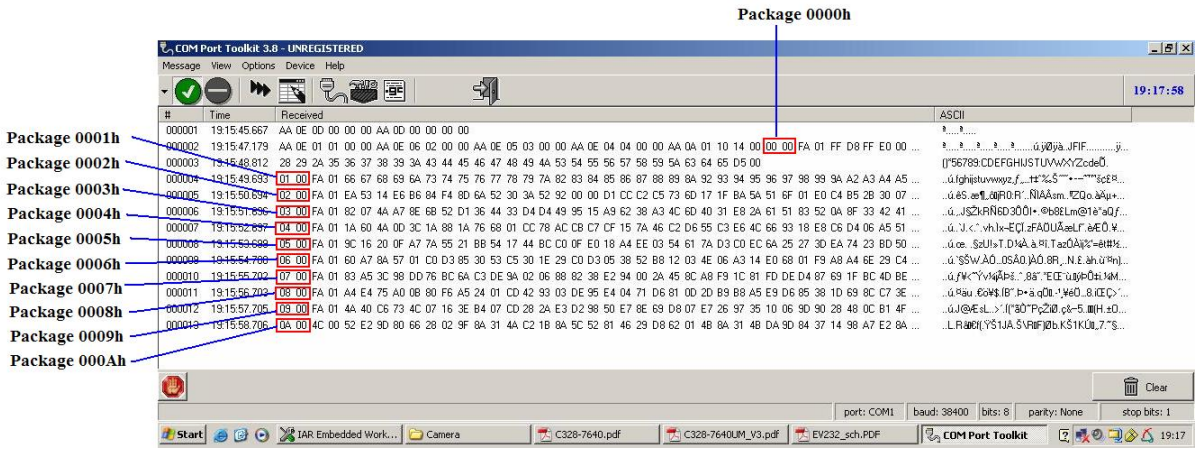


Figure 39: Picture taken by MSP430.

The camera sent a total of 10 packages to the MSP430. The displayed hex data was not yet a picture because it contained package bits. We had to extract the image data from each

package to obtain viewable JPEG picture. Unfortunately, the COM Port Toolkit did not allow us to save the incoming data as a file. We could accomplish this after the completion of the memory and development board integration.

6.3.3 Interface between Memory and Development Board

In this section, we describe the integration between the SD memory card and the development board. The section also discusses the testing procedures, as well as the results, of this integration.

To test this interface, we connected the SD card to a port on the MSP430; this connection required some additional pull-up resistors. Figure 40 and Figure 41 depict the schematic and the actual pieces of hardware, respectively.

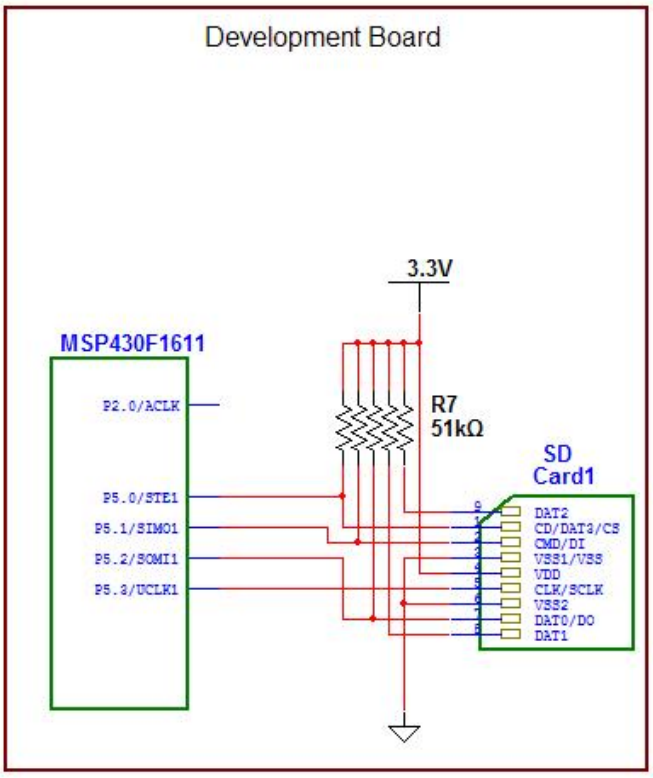


Figure 40: Schematic of SD card and development board.

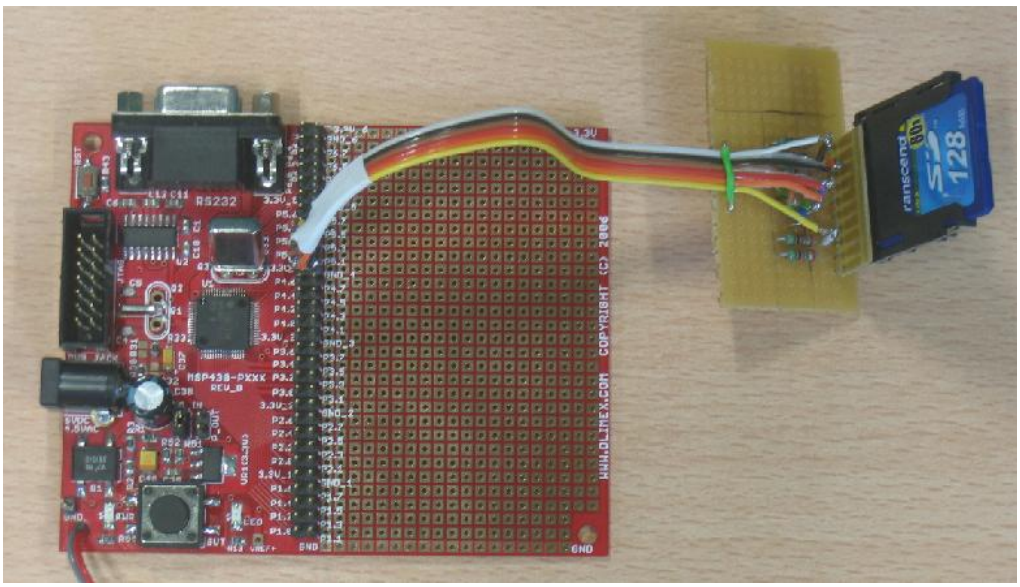


Figure 41: SD card and development board.

SD cards support two major communication modes: SD mode and SPI mode. SPI is a popular synchronous serial protocol for interfacing peripheral devices with many modern microcontrollers, including the MSP430. We implemented the SPI mode in our design for data transmission between the Transcend SD card and the MSP430F1611 chip. In this integration, we configured the card to be the slave device, initiated by the master device—the MSP430 microcontroller. In order to communicate, the MSP430 chip would have to first initialize the card, which would then send back a response. If the initialization was successful, the microcontroller could start transferring blocks of data to the SD card; each block is typically 512 bytes [18].

Due to time constraint, we decided to incorporate code into our program that has already been written. We found a program written by Rolf Freitag to interface between the MSP430 and the SD card. The Lesser GNU public License (LGPL) protects this code, so we are allowed to use it with proper citation. It should be noted that the MSP430 has two SPI ports: SPI0 and SPI1. Freitag's code implemented the SPI1 port, which was compatible with our design. In order to transfer data to the card, we made use of two functions, mmcReadBlock and mmcWriteBlock, respectively. These functions are available in Appendices refcameraNodeCode. These functions allowed us to specify which block of data from which to read and to which to write. After successfully compiling the code, we developed a test to verify this integration.

In this test, we programmed the MSP430 to write several blocks of data to the SD card. The chip then read those blocks and output the data to the computer via UART0 port. Figure 42 illustrates this idea.

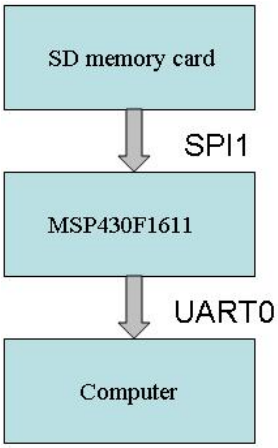


Figure 42: Flow chart of testing SD card with board.

Figure 43 shows the connections among the devices involved in this test.

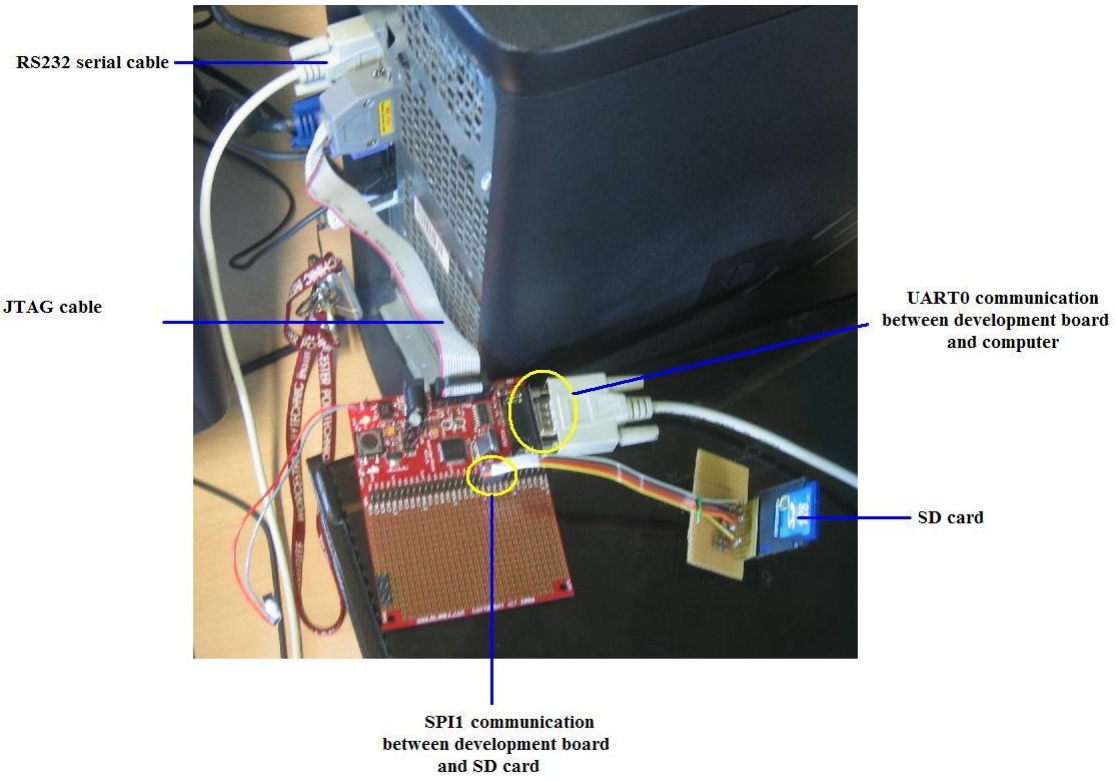


Figure 43: Testing SD card and development board.

Again, we utilized the COM Port Toolkit software to receive and display the data received from the MSP430 microcontroller. The test required the MSP430 to write to the first block of the SD card with a letter 'W', the second block with a letter 'P', the third block with a letter 'I', the tenth block with a letter 'U', and the eleventh block with a letter 'L'. The COM Port Toolkit displayed the binary data it received in the hexadecimal format. In addition, the program displayed the corresponding ASCII value of the data in the ASCII column. In conclusion, this test verified the integration between the SD card and the MSP430 microcontroller.

6.3.4 Interface between Transceiver and Development Board

This section briefly describes the interface between the transceiver and the development board. As mentioned previously in the System Requirements section, the camera node should be capable of transmitting and receiving photos wirelessly. The mesh network team designed the wireless communication portion of this project. They selected the EM260 ZigBee Network Co-Processor from Ember Corporation. It is capable of establishing a ZigBee network for wireless data transmission. The EM260 chip communicated with the MSP430 microcontroller via the SPI0 port. We integrated this ZigBee co-processor into the system so that the images data stored in the SD card could be transmitted either to another camera node or back to the base. Figure 44 illustrates the major tasks of the complete camera node system.

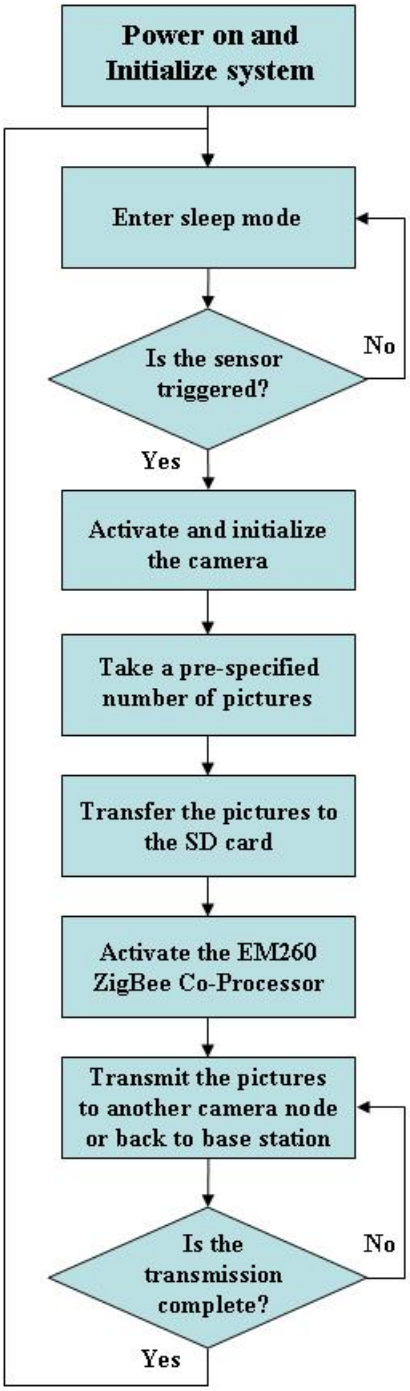


Figure 44: Flow chart of high-level camera node system behavior.

As one can see in Figure 44, the pictures taken by the camera have to be successfully transferred to the SD card so that the EM260 chip could perform its tasks. In short, we had to ensure that our particular design functioned properly.

6.3.5 Interface between Base Station and Mesh Network Team

In this section, we describe the interface within portions of the base station. Once the wireless portion of the base station received the images, the mesh network team programmed

their hardware to save the images a specific directory - `C:\base`. As the website runs on the UL web server, separate from the base station, new image files had to be sent from the base station to the web server. We were not able to control this transfer from the web server, as UL did not support PHP's FTP libraries. Therefore, we chose to initiate this transfer by creating a program to run on the base station to periodically check for new images and transfer them.

As we were most familiar with C++, we chose this language to write the program on the base station. We used the MinGW environment to create and test the program. This program had much the same flow as the scripts for the user interface. The code is available in Appendix H. It periodically obtained directory information and compared it to known images to determine what images were new. Then, it initiated an FTP connection with the web server and sent the new images. After writing this program, we tested it by moving images into the base station directory. The program recognized the new files and successfully sent the files to our share on the web server.

6.4 System Level Testing

In this section, we describe our testing of the system as a two blocks. First, we describe our results when testing the camera node with all its components. Then, we describe the final behavior of the base station.

6.4.1 Camera Node Testing

This section describes the system-level integration of our design. It also presents the camera node system's flowchart, which illustrates the process of taking and storing images. Finally, the section discusses the method for obtaining viewable JPEG pictures from the SD card.

As explained previously in the System Integration section, we were successful in implementing the integrations between each of the following components with the development board: the motion sensor, the camera module, and the SD card. After successfully integrating these components, we performed the overall system integration. As a review of the system, Figure 45 is the schematic of our design.

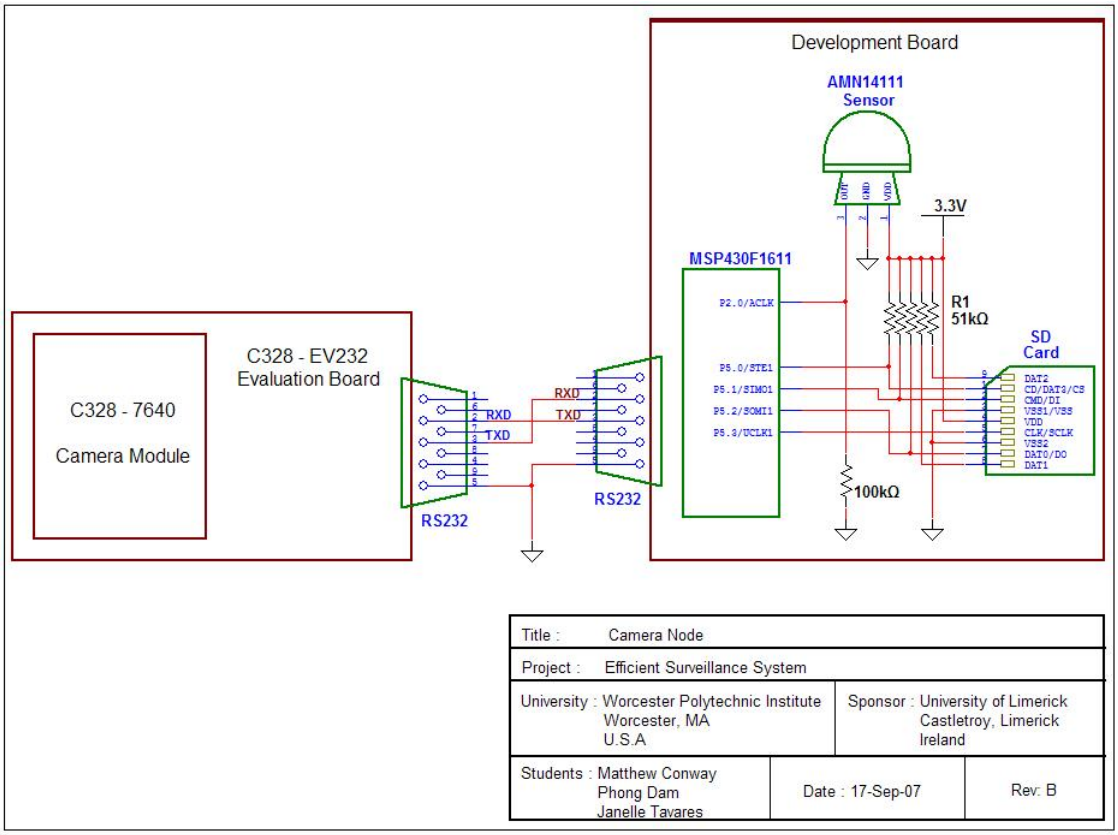


Figure 45: Schematic of the camera node system.

In addition, we created a circuit board with four green LEDs for debugging purpose and monitoring the system's operating status. Figure 46 shows the hardware pertaining to the camera node, as well as the debugging circuit board.

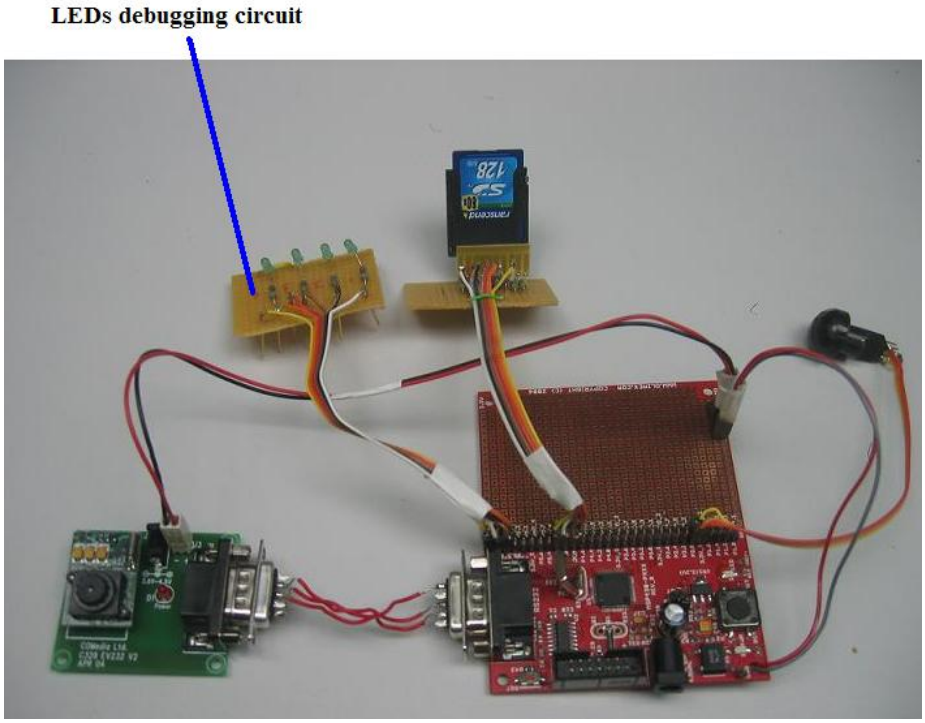


Figure 46: Camera node system with LED circuit board.

With all the components integrated into the system, we list the logical tasks for the system's operation:

1. The MSP430 begins in sleep mode.
2. As soon as the motion sensor detects human's movements, it wakes up the MSP430.
3. The MSP430 immediately activates and requests the camera to take pictures.
4. The microcontroller then requests the camera to send the image data packages. It extracts the actual image data from each package and transfers it to the SD card, prior to the arrival of the next data package.
5. As soon as the transferring process is complete, the MSP430 deactivates the camera and the SD card.
6. After ten seconds of inactivity, the MSP430 enters sleep mode.
7. Steps 2 through 7 repeat.

We used the LED circuit to monitor the status of some of the above tasks. In particular, we specified the LED's indication as follows:

- LED 1 indicates the successful activation and initialization of the camera
- LED 2 indicates the successful transfer of two data packages to the SD card.
- LED 3 indicates the successful transfer of one picture to the SD card.
- LED 4 indicates the successful transfer of a user-specified number of pictures to the SD card. The system enters sleep mode after this LED is lit up.

We created a flowchart for illustrating the routine of the system's operation, which is shown in Figure 47.

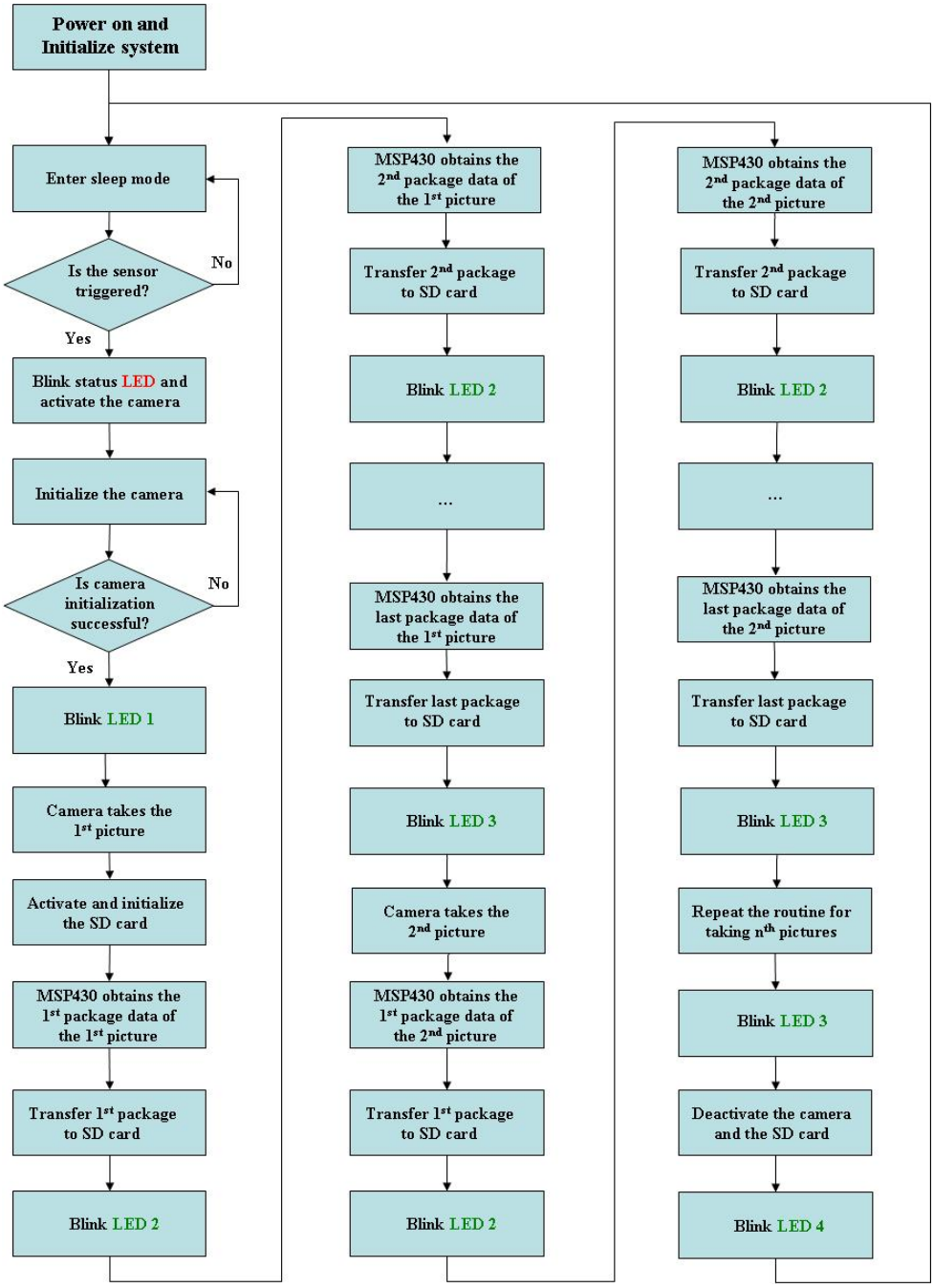


Figure 47: The flow chart of the camera node system.

As one can see in Figure 47, once the sensor wakes the MSP430 microcontroller, the camera can take a user-specified number of pictures with various resolutions. In this design, we programmed the MSP430 chip so that the system took two pictures with a resolution of 640 x 480 when trigged. Figure 48 presents the camera node under system testing.

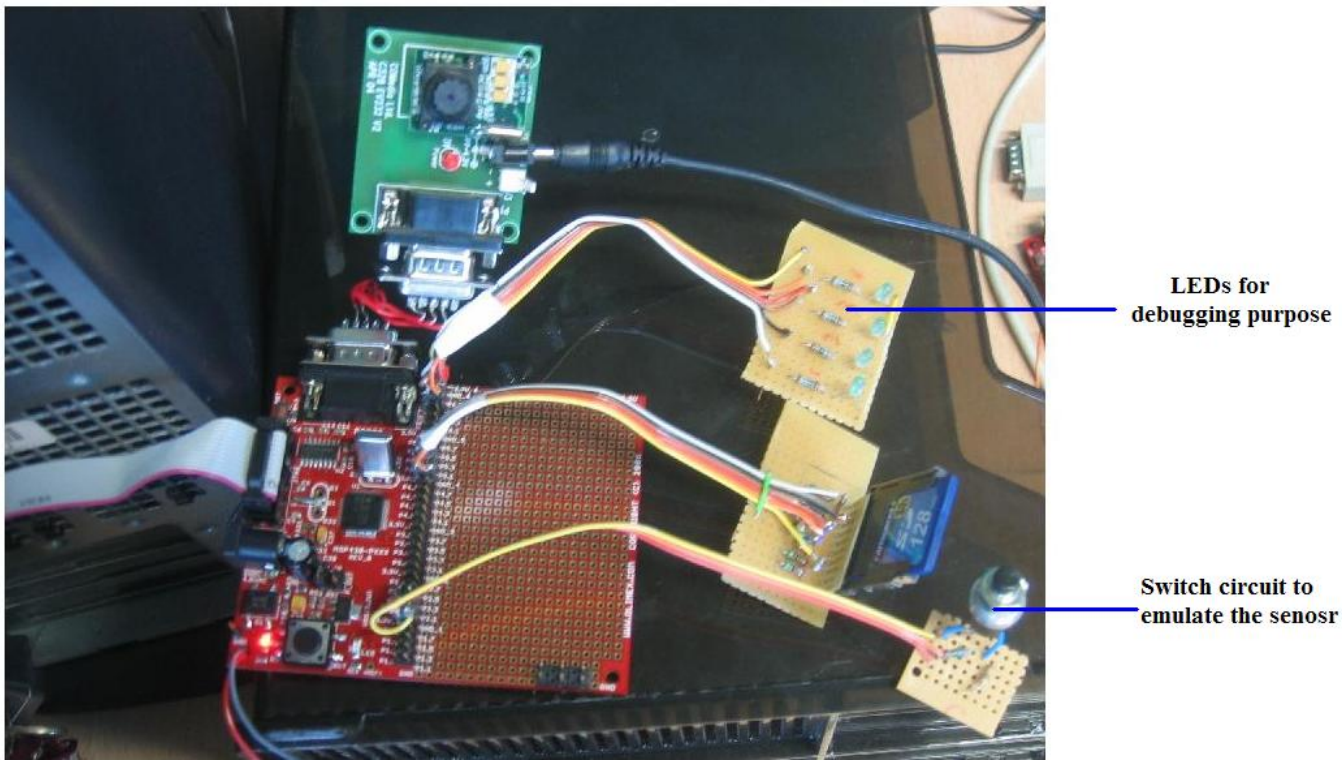


Figure 48: Testing the operation of the camera node.

In Figure 48, we replaced the sensor with a switch circuit, which was described in the Development Board Testing section, to manually trigger the system under testing. We loaded the code onto the MSP430 microcontroller and started testing the system's behavior. After triggering the MSP430, we concluded that the system could operate because the LEDs were blinking. We observed that the LED 4 was lit up; this indicated that the second picture was completely transferred to the SD card. In order to view the data on the card, we repeated the test that was developed for verifying the integration between the SD card and the development board. The MSP430 output the card's data and the COM Port Toolkit program displayed the data on the computer's screen. The camera could take pictures and the MSP430 could transfer the data to the SD card; however, the data on the card was not assured to be JPEG image data. Therefore, we proceeded to perform research on existing software that would allow us to obtain viewable JPEG images from the binary data on the card. We found a commercial program named Advanced Serial Port Monitor (ASPM) from AGG Software, a company that provides software for monitoring and logging incoming data from various peripheral devices via commonly used ports, such as USB and serial COM port. One feature that this ASPM program possesses is the capability of saving data received from the serial COM port as any types of file format, including JPEG file. Figure 49 depicts the user interface of the ASPM program.

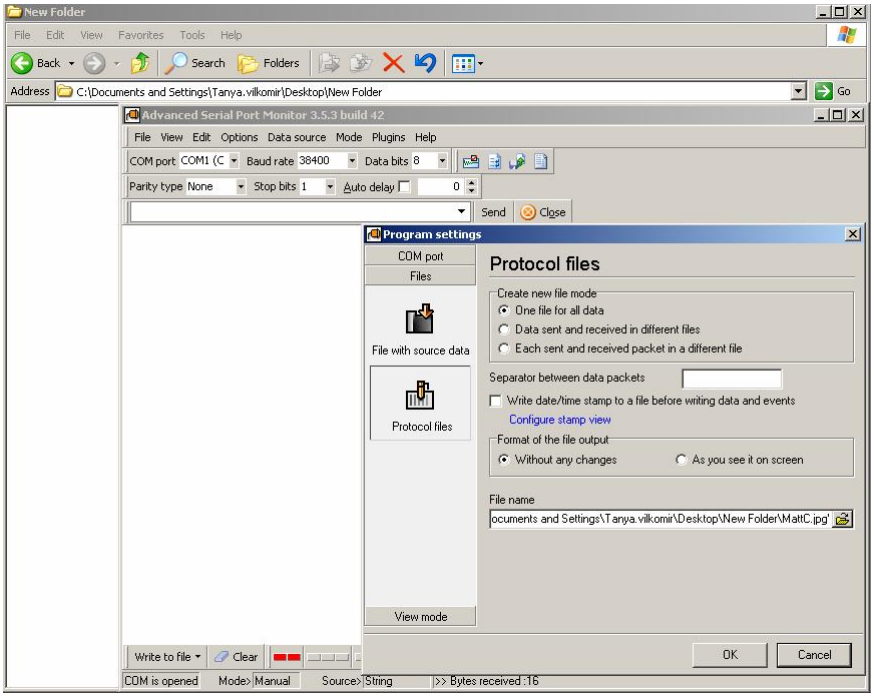


Figure 49: Advanced Serial Port Monitor software user interface.

As one can see in Figure 49, we specified the directory for saving the incoming data as "'MattC.jpg.'" Once the ASPM completely received the data from the SD card, it generated the image file. The captured picture had a resolution of 640x480. The image is available in Appendix E. This process confirmed that the data stored in the SD card was complete and accurate JPEG image data. We evaluated the system based on the speed requirement, i.e. taking 0.5 pictures per second. Table 25 shows the average time for taking pictures with the three different resolutions that the camera supports.

Table 25: Average time (seconds) for taking pictures.

Resolution	**Average size**	**1 picture**	**2 pictures**	**10 picture**	**20 pictures**	**Pics/sec**
160x128	8kB	7.5s	12s	42s	82s	0.24
320x240	14kB	8.5s	16s	75s	145s	0.14
640x480	35kB	15s	34s	165s	310s	0.06

At the smallest resolution, we were only able to achieve half the desired speed; however, this resolution produces unclear and undesirable images. As seen in Table **??**, the higher the resolution is, the slower the system's speed becomes. We achieved these results at a baud rate of 38,400 bps. We attempted but failed to impletement the next fastest compatible baud rate, 115,200bps, between the MSP430 and the camera module. At the fastest baud rate, the system started generating errors after taking a few pictures. In particular, when errors occurred, the

MSP430 stopped transferring data to the SD card; and the status LED on the MSP430 board was blinking unexpectedly. This system behavior repeated after several trials. As a result, we decided to use the 38,400 bps for the camera node system without resolving the aforementioned issue due to time constraint. However, the selected baud rate reduced the system's speed significantly. This led us to perform timing analysis on the camera node system. Prior to examining the timing issue, we created the following list of the major tasks which had an impact on the system's speed:

- Task 1: The MSP430 activates the camera module.
- Task 2: The MSP430 receives a data package, 262 bytes, from the camera module.
- Task 3: The MSP430 writes a block of data, 512 bytes, to the SD card.

In Task 1, the MSP430 continually sends the SYNC command until it receives the ACK from the camera. The microcontroller then issues a sequence of commands to configure the JPEG file format, the resolution, the 38,400 bps baud rate, and the package size of 262 bytes. We found that this task takes about 3 seconds.

In Task 2, the MSP430 receives a data package of 262 bytes. In this package, the actual image data is 256 bytes, and the remaining 6 bytes are for package ID and verify code. In order to send 1 data byte - 8 bits - in UART communication, a start bit and a stop bit are added for each data byte. Therefore, the total bits of each package can be calculated as follows:

$$(1stopbit + 8databits + 1stopbit) \cdot 262bytes = 2620bits$$

Using the above result, we computed the duration for the MSP430 to receive one package as below:

$$\frac{2620bits}{38400bps} = 0.0682s$$

In Task 3, the MSP430 writes a block of 512 bytes to the SD card once it receives completely 2 packages from the camera. It should be noted that there are 256 bytes of actual image data in each package. Thus, the MSP430 will combine the actual image data of 2 packages prior to transferring to the card. We implemented a clock speed of 2MHz for this SPI communication between the microcontroller and the card. In SPI mode, the MSP430 sends a bit every clock cycle. This indicates that the speed of this SPI interface is 2,000,000 bps. Moreover, there is no additional bit which is sent with each data byte. Therefore, the total bits that the MSP430 transfers to the card every block-write is calculated as follows:

$$512bytes \cdot 8bits = 4096bits$$

We used the above number of bits to compute the duration for each block-write as below:

$$\frac{4096bits}{2,000,000bps} = 0.0020s$$

In general, Task 1 is executed after the sensor triggers the system. Task 2 and Task 3 are repeated until the MSP430 receives and transfers all the packages to the SD card. As discussed

previously, we programmed the system to take 2 pictures with a resolution of 640 x 480. Referring to Table 25, the average size of a 640 x 480 resolution picture is 35kB. This generates roughly 136 data packages and 68 block-writes to the SD card. Using these values, we estimated the time T for taking and transferring completely one picture to the memory card.

$$T = (136 packages \cdot Task2 duration) + (68 block - writes \cdot Task3 duration)$$

$$T = (136 \cdot 0.0682s) + (68 \cdot 0.0020s)$$

$$T = 9.2752s + 0.1360s$$

$$T = 9.4112s$$

Table 26 shows the theoretical time for the entire process of taking 2 pictures.

Table 26: Theoretical time for taking pictures at 38,400 bps.

	Duration (seconds)
Task 1	3.0000
First picture	9.4112
Second picture	9.4112
Total	21.8224

However, we did not yet take into account the duration for blinking the debugging LEDs which were used for obtaining the results in Table 25. As mentioned previously, we had four green LEDs to monitor the status of the system. Each LED was configured to last for a short duration of 0.05s. Based on the LEDs' indication defined in the program, we created Table 27 to show the number of blinks for each LED and the total duration for blinking the LEDs when the system takes and stores 2 pictures.

Table 27: LEDs blinking duration.

LED	**Blinks**
1	1
2	136
3	2
4	1
Total	140
Duration	7s

In order to compare the theoretical time to the measured time of taking 2 pictures, the LEDs' duration should be added to the result in Table 26. Thus, the total theoretical time is about 28.8 seconds which is 15.3% less than the measured time, 34 seconds. This percentage

difference can be due to error in measurement. Moreover, the theoretical time only took into account the major tasks' duration. The MSP430 does require considerable time to process data, i.e. writing data to the transmit buffer prior to transfer to the SD card. Unfortunately, we were unable to obtain this information. Hence, the theoretical result can not adequately address the timing issue; but, it still serves as a useful tool to evaluate the performance of our system.

Similarly, we performed timing analysis for the fastest baud rate, 115,200 bps, between the camera and the MSP430. Table 28 shows the estimation.

Table 28: Theoretical time for taking two pictures at 115200 bps.

	Duration (seconds)
Task 1	3.0000
First picture	3.2232
Second picture	3.2232
Total	9.4464

As one can see in Table 28, if the fastest baud rate had been successfully implemented, this could have greatly improved the speed of the system. Furthermore, the debugging LEDs should be disabled in actual operation in order to eliminate the unnecessary delay. In conclusion, aside from the system's speed issue, all other aspects of integrating the camera system were successful.

6.4.2 Base Station Testing

In this section, we describe the implementation and results of the base station. We describe the final behavior of the base station code, as well as the results of testing.

The functions of the user interface of the base station were displaying all photos taken by the surveillance system and notifying the specified users of new activity. We accomplished the functionality of the base station with PHP scripts, embedded into a simple website that we made with HTML. The code performed these tasks by first checking our website directory on UL's web server for new pictures. If it found new pictures, it added them to a list of known images and uploaded them to the website. Also, if the code found new images, it notified the list of individuals of this new activity, as well as the list of mobile phones. The website refreshes itself every thirty seconds to repeat this process. Figure 50 describes this behavior in a flow chart.

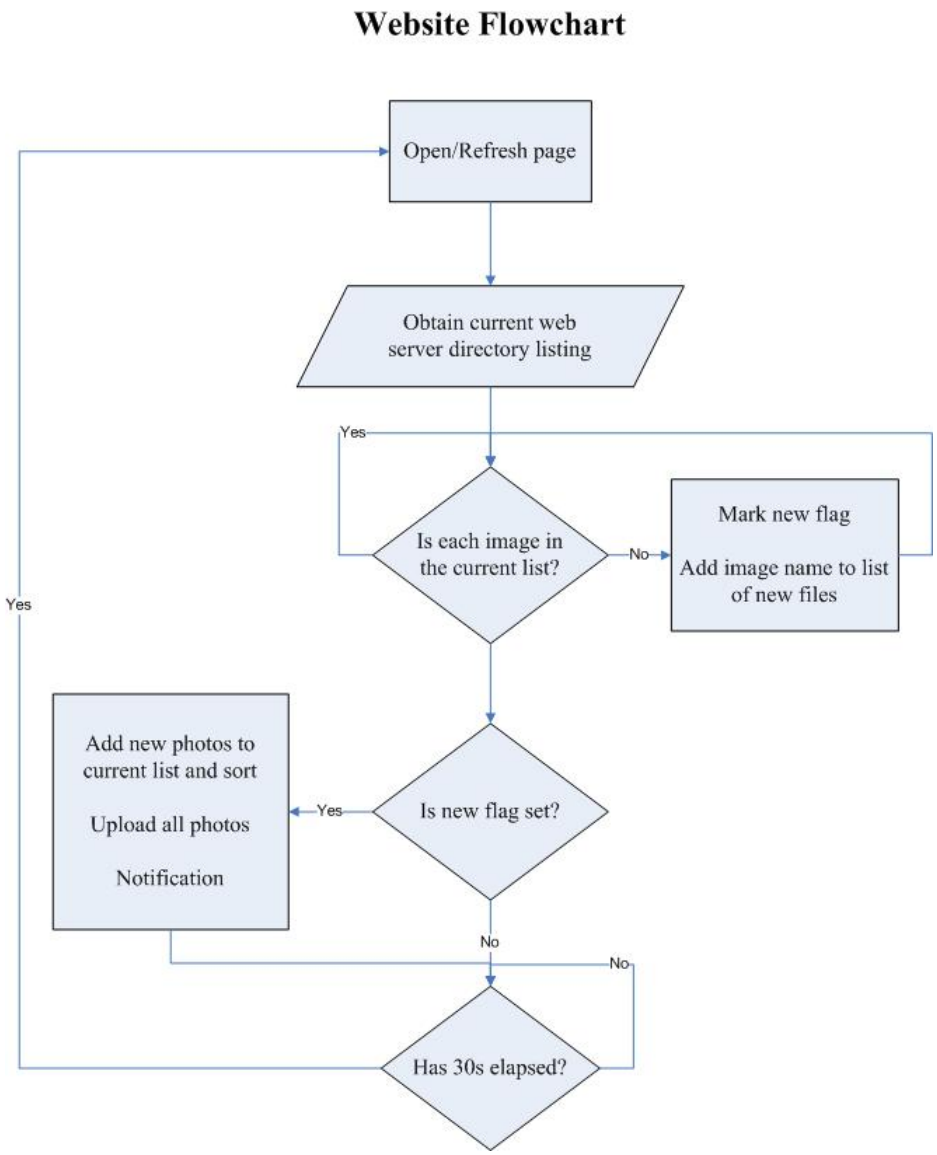


Figure 50: Flow chart of website behavior.

The code to implement these functions is available in Appendix H. We tested these scripts by opening the website in a browser. As UL hosted our website, our website address was http://www.ul.ie/ULMQP07. The site appears in Figure 51.

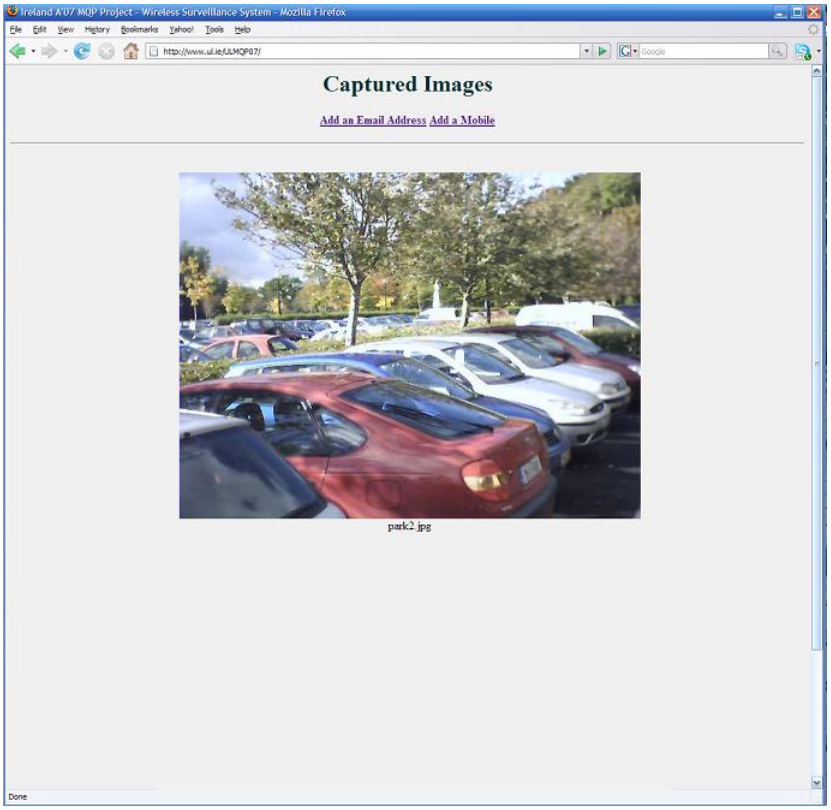


Figure 51: Screen shot of the base station website.

On the base station, we ran a program written in C++ to recognize these new images. In addition, once this program recognized new images, it would send them to our share on the web server using FTP. As the microcontroller in the mesh network team's portion of the base station moved new images into the specified directory, the program recognized the new images and sent them. Then, the PHP scripts uploaded the new image and sent alert messages after refreshing the page. These messages included a text message and an email with an attachment of the new image. Figure 52 and Figure 53 show receipt of these alert messages.

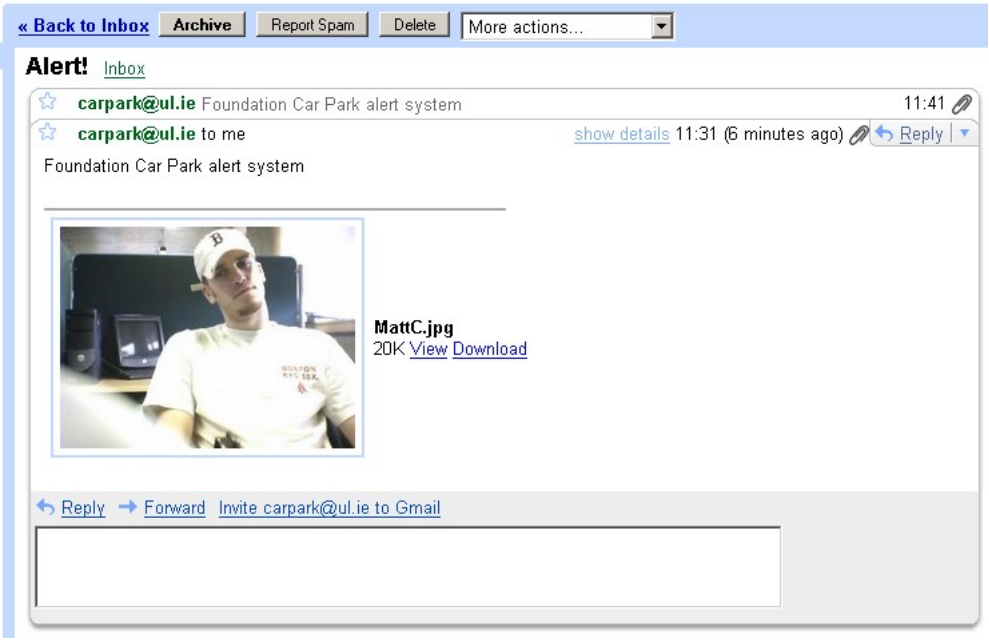


Figure 52: Alert message via email.

Figure 53: Alert via text message.

7 Conclusions

In this chapter, we summarize the results of our design of a low-cost, low-power wireless surveillance system. We review the requirements and describe how our results compare to those requirements. For convenience, we provide the requirements again.

- Camera node
  - *Ability to detect movements:* A motion sensor would be used for detecting movements.
  - *Ability to take photos:* A digital camera should be able to take still photos as soon as activity triggers the motion sensor. Upon that trigger, the camera should take two pictures per second for five seconds.
  - *Storage for taken photos:* The storage should be a memory device that can hold the photos temporarily or permanently.
  - *Battery powered and energy efficiency:* The sponsor desired the system to be wireless, the camera node should operate efficiently on battery power; the battery should last at least for a year.
  - *Central controlling unit:* This unit would control the entire operation of the camera node.
- Base station
  - *Storage for receiving photos:* This storage should be a computer hard drive and its capacity should be large enough to store the photos received from the camera nodes.
  - *Ability to upload photos to website:* We should design a website for uploading the photos from the hard drive.
  - *Ability to send messages or photos to mobile phones:* Notifications should be sent to users' mobile phones of activity in the car park.

For the camera node, we were able to sense motion, take images, and store images. The PIR sensor was able to indicate to the MSP430 when there was motion within its range. This sensor had a sensitivity range of about 10m at angles ranging from 124° to 34° . The MSP430 was able to process this signal, as well as control the camera and communicate with the SD card. On the base station, we were able to store images, upload them to a website, and send alert messages. The base station had sufficient storage to store images. The base station programs sent new images to the web server, then uploaded images to the website and notified the user-specified individuals of new activity.

In total, our final system met all but two requirements. The first area in which our design fell short was the rate of pictures taken after sensing motion. We were able to take two pictures with 640x480 resolution in 34 seconds after the PIR sensor triggered an interrupt on the MSP430. We pointed out in Section 6.4.1 that the measured slow speed was due to the implementation of the

low baud rate, 38,400 bps, and due to the delay of blinking the debugging LEDs. However, even if we had been successful implemented the fastest baud rate, 115,200 bps, between the camera and the MSP430, we still could have not met the image capturing rate requirement.

The second requirement that we did not meet was the lifetime of the power supply. Though we estimated that the battery could be usable for several years, there are certain months of the year in which the amount of sunshine is insufficient for the solar panel to keep the battery charged. Refer to Table 23 for these calculations. With our design, the users of this system would have to maintain the power supply by fully charging the battery about three times during the winter.

As an overall goal of being less costly than a traditional car park security system, we found that our system would be about five times more effective. Refer to Appendix B for the calculations of our prototype when produced in bulk. The cost of raw materials for a camera node and base station per 1000 quantity were €171.53 and €88.98, respectively. Installation would consist of mounting each node onto light posts already located in car parks, and this would factor into the overall cost as labor. These figures are considerably smaller than the quote we were given, €3,298 for a system with four cameras. For four cameras, our system would cost about €775, without installation. For a figure closer to the quote we received, our system with twenty camera nodes would cost about €3,500.

In summary, we met most of the requirements for this system. In our recommendations, we convey our ideas on changes in order to meet these requirements, as well as other improvements on this wireless surveillance system.

8 Recommendations

We were successful in implementing and testing a proof-of-concept device for our portion of the low-power, wireless surveillance system; it met most of the basic requirements of this project. To be a marketable product, the device would require more work. This chapter describes our recommendations for future work on this prototype. It first contains our recommendations for meeting all requirements, then recommendations about preparing the prototype for market, and finally any additional recommendations that we felt were noteworthy.

8.1 Recommendations for Meeting System Requirements

At the end of this project, we failed to meet two requirements. We projected that the power supply could not last a full year as several months do not have enough solar insolation to recharge the battery. Also, our camera module was not able to take pictures at a rate of two per second; it was significantly slower. This section describes our recommendations for necessary future work so that this prototype could meet all requirements.

8.1.1 Meeting the Image Capturing Rate Requirement

We were able to interface the camera module and MSP430 with a UART interface. While the processing time of the image data was a factor, one of the noticeable bottlenecks in this portion of the system was the UART interface. At a baud rate of 38,400bps, we were able to successfully configure this interface. This rate is the second fastest rate that is compatible with both devices; the fastest rate is 115,200bps. As discussed in Section 6.4.1, we failed to implement this fastest rate which could greatly improve the speed of the system. While we are not sure if two pictures per second is an attainable rate while using most of the components we chose, we recommend that teams doing future work on this project should implement this maximum baud rate. Another solution could be using a camera module which supports SPI interface in order to meet the capturing rate requirement. Furthermore, using a camera module with this interface would require the team to find another microcontroller which has additional SPI ports, as we already used all that were available in our design.

8.1.2 Meeting the Battery Life Requirement

In theory, we found that our power supply would not be able last the required length of time, one year. The operators of this system would have to remove the batteries for charging about three times over the course of the winter for the power supplies to continually power the system. We made these calculations assuming that the camera nodes would be on for eight hours during the night. As we did not have a means to turn the camera nodes off and on during the other 16 hours of the day, the actual power consumption with this system would be greater than these figures when taking into account this sleep mode consumption. Our recommendations for

meeting this requirement center around making the charging method more effective and making the camera node more power efficient.

In order for the battery to last through the winter, the charging method must be more effective during these months. We recommend that future teams that work on this project consider more powerful or larger solar panels for the charging method. In addition, if solar panels prove not to be efficient during the winter in Limerick, we recommend that future teams consider adding additional methods of harvesting energy from the environment, such as temperature gradients, vibrations, and ambient radiation

At the end of the project, the system's sleep mode consisted of turning off the camera module after a period of inactivity. The PIR sensor had to be able to detect motion at any time, and the transceiver had to be able to route traffic. Though the microcontroller received power, it ran in its low-power sleep mode. Concerning the state of the system during the day, it would have to remain in sleep mode with a deactivated interrupt, continuing to consume a small amount of power. To be even more energy efficient, all devices could be turned off during the day when monitoring the car park is not necessary—a "'deep sleep'" mode. One could accomplish this mode by using another interrupt on the microcontroller and a timing circuit. The circuit would need to send a signal to initiate the microcontroller at a user-defined time of the day. Another signal after about eight hours would indicate to the microcontroller to turn everything off, including itself. For instance, the surveillance system may only need to be active after sunset and before sunrise, so these times could be used as the initialization and deactivation times for this additional deep sleep mode. The only power consumption, which would be very small, would be the timing circuit, much like a PC while it is powered off. This mode would help the camera node consume less energy and would help extend the lifetime of the battery to meet the requirement.

8.2 Recommendations for a Marketable Prototype

In this section, we describe our recommendations to prepare this prototype for market. These recommendations center around improving the basic functions of the system.

8.2.1 Recommendations for the Camera Module

At the completion of this project, the Panasonic AMN14111 PIR sensor was able to detect motion in areas that were out of the range of the camera. The camera module would be more effective if its horizontal and vertical ranges were more comparable to those of the PIR sensor. Figure 54 illustrates the car park from above with the horizontal ranges of our image sensor and PIR sensors that we found in their documentation. That is, the figure displays the horizontal range of the PIR sensor (shown in yellow, green and blue) as about 110° at 10m, while the camera range is only 42° (shown in violet). Figure 55 illustrates from the side the vertical range of the PIR sensor (shown in violet) and the vertical range of the camera (shown in red, green and blue as if mounted on a lamp post at various heights). In this image, we chose to display

the range of the camera at several heights to show the different levels of coverage. We plotted the ranges such that they can keep a 2m-tall person in full view while in the middle of a car park lane. In addition, the figure shows at what distance from the lamp post that the range would no longer be able to capture any portion of a 2m-tall person. We used the values from the documentation; the vertical range of the PIR sensor is about 93° at 10m, while that of the camera is only 16.5°.

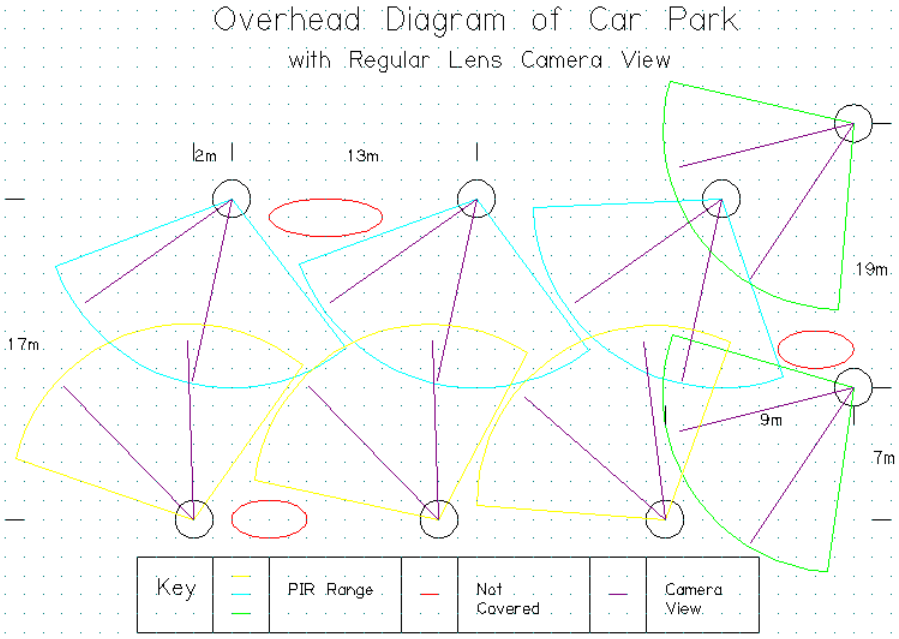


Figure 54: Overhead view of car park with ranges for PIR sensor and camera.

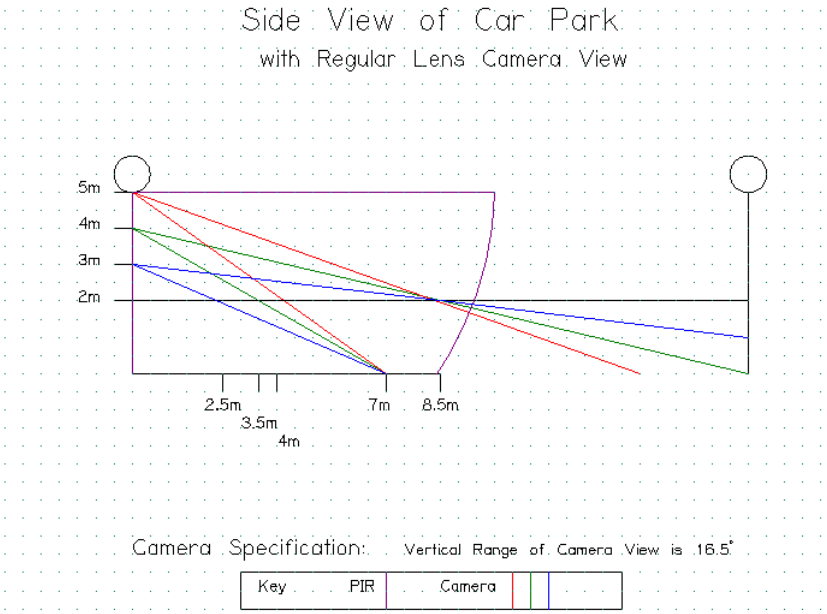


Figure 55: Side view of car park with ranges for PIR sensor and camera.

One can see from these diagrams that some areas covered by the PIR sensor are not in range of the camera. We recommend that future work on this system include adding a wide angle lens with the camera module. Using a wide angle lens, the range of the camera would be more comparable to the PIR sensor, and blind spots could be reduced or eliminated. One suitable lens for our $\frac{1}{4}$ in. format camera is the 2.3mm F2.5 High Resolution CCTV Lens from Marshall Electronics. It has a horizontal range of 91.9° and a vertical range of 69.1°, as compared to 42° and 16.5°, respectively [53]. Figure 56 illustrates this additional range on the horizontal, and Figure 57 illustrates the vertical range, at various heights. With a wide angle lens, the visual range of the camera would be closer to the range of the PIR sensor.

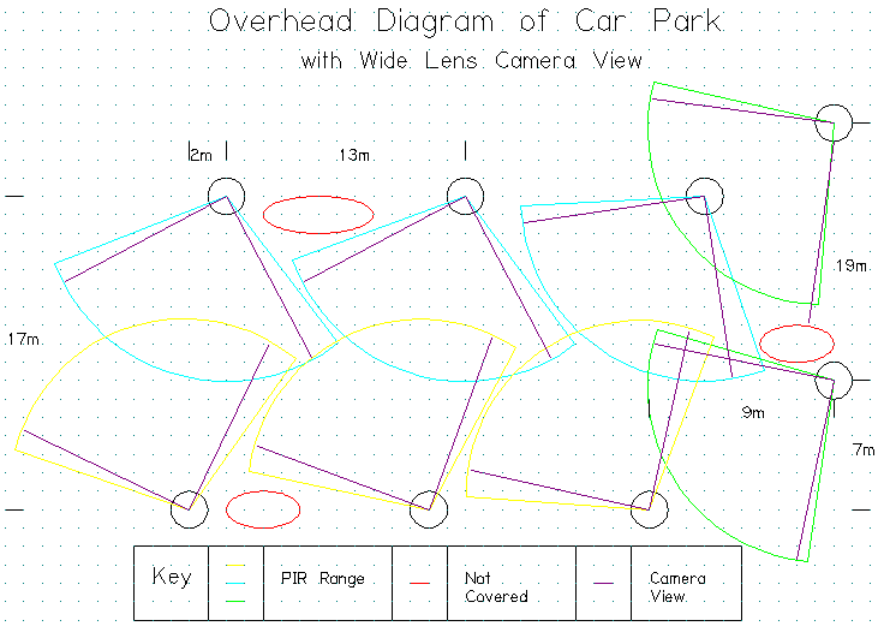


Figure 56: Overhead view of car park with ranges for PIR sensor and camera.

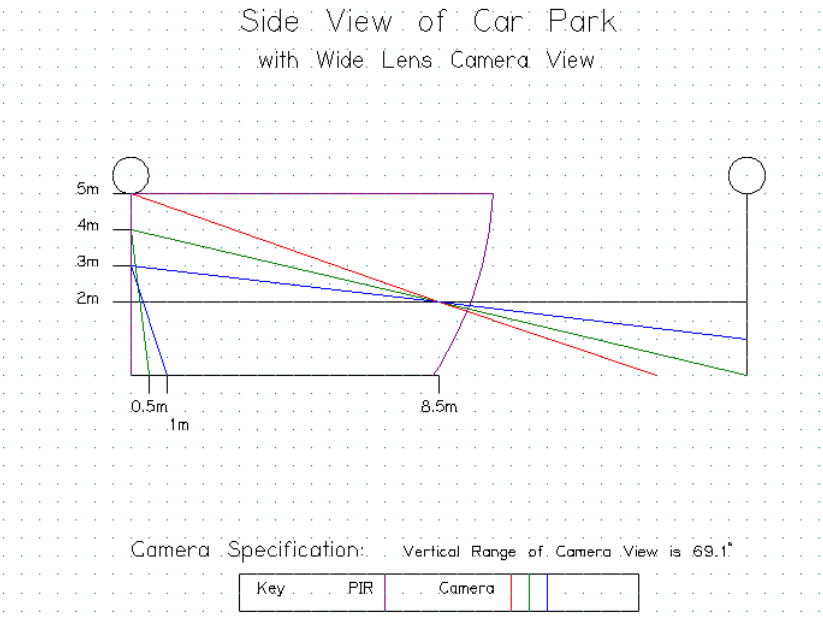


Figure 57: Side view of car park with ranges for PIR sensor and camera.

8.2.2 Recommendations for the PIR Sensor

Our recommendation for the motion sensing portion of this system centers around making the system cheaper for the customer. In addition, implementing this recommendation would make this portion of the system more effective.

At the end of the project, the PIR sensor's effective range was at most 10m. The range of the camera depended on the resolution, more than anything else, and could potentially gather information much further than 10m. Also, with the set up suggested in the figures in Section 8.2.1, two nodes would have to be mounted facing opposite each other on most lamp posts. Requiring this number of camera nodes would be quite costly for larger systems. Using a PIR sensor network would help to minimize the number of camera nodes necessary to cover the same area, while taking advantage of the viewing range of the camera. The network would consist of several PIR sensors that communicate wirelessly with the camera node. This network would be set up to sense motion in the range of the camera; in this way, the installation of this system would include a camera node and an additional PIR sensor on each lamp post; this installation would involve fewer components than two full camera nodes and, therefore, would be cheaper. In addition, depending on the configuration of the area in which the surveillance system was installed, multiple PIR sensors could cover the same area at multiple angles, reducing the blind spots in the motion sensing coverage.

8.2.3 Recommendations for the Base Station

This section describes our recommendations for preparing the base station for market. We recommended improving the user interface by adding features to the website and reducing the maintenance.

At the end of the project, the base station was able to show the user the captured images and to notify specified individuals of these new pictures arriving at the base station. In addition, one could add addresses to be notified. To improve this interface for the end user, we recommend adding certain features. As a security feature, this website should require employees to log in before accessing images. To improve how a user views images, we recommend adding features on the website such as a means to search for photos, sort them by date or by camera, and add comments or score images to indicate the importance of the information in them. To reduce maintenance, the base station should automatically move old images to another directory to archive them. Also for convenience, the base station should automatically notify the on-duty security personnel based on the time and the date. One could accomplish this with a database of employees, their contact information and their schedules. Finally, if security staff saw suspicious activity in the images captured, they should be able to request a sequence of images from a particular camera for a duration of time after the original image was taken to ensure that the criminal activity would be documented.

Another concern for a product ready for market was resistance to the elements. As these devices would be mounted in relatively open areas on a lamp post, the enclosures for the power source and camera node should be weather proof to protect the components; however, the enclosure for the power supply should allow for some air flow so that is does not overheat.

8.3 Additional Recommendations

This section describes our final recommendations for this security camera surveillance system. These recommendations are for adding features which are not related to meeting the system requirements and which a customer may not require, but may be attractive features.

8.3.1 Recommendations for the Power Supply

To ease maintenance for the customer, we recommend that future groups implement a means to indicate to the users when the battery is low. As it stood at the end of the project, the customer would have to realize that no pictures were received from a certain location before considering that a battery may be completely drained. This indicator would consist of hardware to compare the current voltage in the battery to the minimum acceptable level. We recommend that this hardware be connected to the microcontroller so that it can process this data. The microcontroller should generate an alert message to send back to the base station when the data indicates the battery has approached an unacceptable level. In addition, if the microcontroller accumulates this data over time, the base station could use it to generate a report on the performance of the security system over time. In this way, a user could anticipate, based on trends in the data, when battery would have to be charged or replaced.

Furthermore, we recommend that teams consider adding components to the power supply to control when it charges. The number of charge cycles is a factor in the lifetime of a battery and minimizing the number of charge cycles would extend that lifetime. During parts of the year when the sun is brighter, the battery would need to charge less often to be able to power the circuit continuously. By using a device like a relay, the charge controller and solar panel could be disconnected from the battery to prevent charging everyday. During darker months, the charge controller and solar panel should always be connected. One could control when the battery charges based on the day of the week and month or by creating an algorithm that would be more precise. This algorithm could determine when the battery had to charge based on mAh and data gathered by an ammeter. In addition, it could anticipate a good time to charge and for how long based on the intensity of sunlight, using data from a light sensor.

8.3.2 Recommendations for the Base Station

Our final recommendation is for the format by which the base station notifies individuals' mobile devices of new activity in the car park. Receiving images on mobile devices is convenient and would require the use of an MMS server. As a marketable product, the cost of this service would be worth the attractiveness of the feature to the customer; we recommend future teams that work on this project consider this service.

References

[1]

[2] "2.4ghz Mini Wireless Color Camera," Securityman.com, Table 1. [Online]. Available: http://www.securitymaninc.com/product/zt809t.html. [Accessed: 18 Aug, 2007].

[3] H. Aghajan and S. Hengstler, "WiSNAP: A Wireless Image Sensor Network Application Platform," In Proc. of 2nd Int. IEEE/Create-Net Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities, Mar. 2006, Barcelona, Spain. [Online] Available: http://wsnl.stanford.edu/papers/TridentCom_2006.pdf. [Accessed: 25 Aug 2007].

[4] "AlgoCam^{TM} TC6030," Transchip. [Online]. Available: http://www.transchip.com//objects/TC6030-PB-30March05.pdf. [Accessed: 28 Aug. 2007].

[5] "Alkaline Battery," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Alkaline_battery. [Accessed: 15 Aug. 2007].

[6] M. Barr, "Introduction to Memory Types," Embedded.com. 1 May 2001. [Online]. Available: http://www.embedded.com/columns/beginerscorner/9900121?_requestid=1038792. [Accessed: 10 Aug. 2007].

[7] B. Bonfield and L. S. Quinn, "Comparing Open-Source Content Management Systems," TechSoup.org, 2 Jan. 2007. [Online]. Available: http://www.techsoup.org/learningcenter/webbuilding/page6047.cfm. [Accessed: 16 Aug. 2007].

[8] I. Buchmann, "Can the lead-acid battery compete in modern times?" Batteryuniversity.com. July 2003. [Online] Available: http://www.batteryuniversity.com/partone-6.htm. [Accessed: 10 Sept. 2007].

[9] "C238-7640 JPEG Compression VGA Camera Module," Electronics123.com. [Online]. Available: http://www.electronics123.net/amazon/datasheet/C328-7640.pdf. [Accessed: 9 Sept. 2007].

[10] "C238-7640 User Manual," Electronics123.com. [Online]. Available: http://www.electronics123.net/amazon/datasheet/C328-7640UM_V3.pdf. [Accessed: 9 Sept. 2007].

[11] "C238 Enhanced JPEG Module," Electronics123.com. [Online]. Available: http://www.electronics123.net/amazon/datasheet/C628.pdf. [Accessed: 9 Sept. 2007].

[12] A. Chandrakasan, J. Goodman, J. Kao, W. Rabiner, and T. Simon, "Design of a Low-Power Wireless Camera," MIT. 2004. [Online]. Available: http://ieeexplore.ieee.org/iel4/5411/14614/00667109.pdf. [Accessed: 25 Aug 2007].

[13] Council of Europe, *European sourcebook of crime and criminal justice statistics - 2003*, 2nd ed., ser. Onderzoek en beleid. Meppel: Boom Juridische uitgevers, 2003.

[14] "Crime Statistics: Car Thefts by Country," Nation Master, Seventh United Nations Survey of Crime Trends and Operations of Criminal Justice Systems. [Online]. Available: http://www.nationmaster.com/graph/cri_car_the_percap-crime-car-thefts-per-capita. [Accessed: 6 Oct. 2007].

[15] D. P. Curtin, "A Short Course Book Sensors, Pixels and Image Sizes," Short Courses, 2007. [Online]. Available: http://www.shortcourses.com/sensors/sensors1-0.html. [Accessed: 10 Sept. 2007].

[16] "DCS-5300G > High Speed 2.4GHz (802.11g) Wireless Internet Camera," D-Link, Table 1. [Online]. Available: http://www.dlink.com/products/resource.asp?pid=342&rid=1230&sec=0. [Accessed: 18 Aug. 2007].

[17] "Dynamic random access memory," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Dynamic_random_access_memory. [Accessed: 17 Aug. 2007].

[18] F. Foust, "Application Note - Security Digital Card Interface for the MSP430," Michigan State University, 2004. [Online]. Available: http://www.cs.ucr.edu/~amitra/sdcard/Additional/sdcard_appnote_foust.pdf. [Accessed: 20 Sept. 2007].

[19] "How do motion sensing lights and burglar alarms work?" Howstuffworks.com, para. 3. [Online]. Available: http://computer.howstuffworks.com/question238.htm. [Accessed: 13 Aug. 2007].

[20] "Jpeg," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/JPEG. [Accessed: 17 Aug. 2007].

[21] E. Köppe, A. Liers, H. Ritter, and J. Schiller, *Low-Power Image Transmission in Wireless Sensor Networks using ScatterWeb Technology.* Berlin, Germany: Institute of Computer Science, 2004, [Online] Available: http://www.broadnets.org/2004/workshop-papers/Basenets/Schiller.pdf. [Accessed: 25 Aug 2007].

[22] W. Lee, "Method and apparatus for detecting direction and speed using PIR sensor," U.S. Patent, para. 2, 1 Mar. 1994. [Online]. Available: http://www.google.com/patents?hl=en&lr=&vid=USPAT5291020&id=yAooAAAAEBAJ&oi=fnd&dq=pir+sensors. [Accessed: 13 Aug. 2007].

[23] "Limerick, Ireland—Sunrise, sunset, dawn and dusk times for the whole year," GAISMA, [Online]. Available: http://www.gaisma.com/en/location/limerick.html. [Accessed: 28 Sept., 2007].

[24] "Lithium Ion Battery," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Lithium_Ion_Battery. [Accessed: 16 Aug. 2007].

[25] "Lithium Ion Energy," Rathbone Energy. [Online]. Available: http://www.rathboneenergy.com/batteries/lithium_ion/lithium.htm. [Accessed: 15 Aug. 2007].

[26] T. Markvart, "Solar Electricity," John Wiley and Sons, 2000. [E-book]. Available: Google Books. [Accessed: 4 Sept. 2007].

[27] "Memory Technology Types," The PC Guide. [Online]. Available: http://www.pcguide.com/ref/ram/types.htm. [Accessed: 20 Aug. 2007].

[28] "Mp Motion Sensor," Farnell.com. [Online]. Available: http://www.farnell.com/datasheets/36210.pdf. [Accessed: 9 Sept. 2007].

[29] "Nickel-cadmium Battery," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Nickel_cadmium_battery. [Accessed: 14 Aug. 2007].

[30] "Nickel metal hydride battery," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Nickel_metal_hydride_battery. [Accessed: 14 Aug. 2007].

[31] "Non-volatile memory," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Non-volatile_memory. [Accessed: 17 Aug. 2007].

[32] "OV630," Omnivision. [Online]. Available: http://www.ovt.com/products/ip_detail.asp?id=22. [Accessed: 10 Sept. 2007].

[33] "Panasonic Network Camera Network Camera Package," Amazon.com. 25 Nov. 2006. [Online]. Available: http://www.amazon.com/gp/product/B000LSIY54/ref=nosim?s=merchant&m=A2Y624Y5TXKSKI&v=glance&tag=nextag-ce-tier6-20&creative=380341&linkCode=asn&creativeASIN=B000LSIY54. [Accessed: 14 Aug. 2007].

[34] "A Performance Calculator for Grid-Connected PV Systems," Renewable Resource Data Center. [Online]. Available: http://rredc.nrel.gov/solar/codes_algs/PVWATTS/version1/. [Accessed: 12 Sept. 2007].

[35] "Perl ASP VBscript PHP JSP Comparison: Comparing Web Scripting and Website Programming Languages," Gbdirect. [Online]. Available: http://training.gbdirect.co.uk/courses/php/comparison_php_versus_perl_vs_asp_jsp_vs_vbscript_web_scripting.html. [Accessed: 18 Aug. 2007].

[36] K. Pointer and N. Tilley, in *A Review of CCTV Evaluations: Crime Prevention Effects and Attitudes Toward Its Use*, C. Phillips, Ed. New York: Criminal Justice Press, 1999, ch. Surveillance of Public Space, pp. 30–33.

[37] "Provideo Wc 2503 Wireless Color Weatherproof Ir Camera," Shopping.com, para. 1. [Online]. Available: http://www1.shopping.com/xPF-CSi-Speco-Provideo-Wc-2503-Wireless-Color-Weatherproof-Ir-Camera.[Accessed: 17 Aug. 2007].

[38] "SecurityMan Mini Airwatch 4 System," Cu1.com, para.1-3. [Online]. Available: http://www.cu1.com/semiai4sy.html. [Accessed: 16 Aug. 2007].

[39] Síochána, *Annual report - Garda Síochána.* Dublin: Stationery Office, 2005.

[40] ——, *Annual report - Garda Síochána.* Dublin: Stationery Office, 2006.

[41] "Small Wind Turines," Mayo Energy Agency, Ltd. [Online]. Available: http://www.mayoenergy.ie/Home2/Publications/PDFFile,2545,en.pdf. [Accessed: 3 Sept. 2007].

[42] "Static random access memory," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Static_random_access_memory. [Accessed: 17 Aug. 2007].

[43] "Transcend releases its newest 80X SD Card featuring outstanding data transfer speeds," TranscendUSA.com, 15 Apr. 2005. [Online] Available: http://www.transcendusa.com/Press/index.asp?LangNo=0&axn=Detail&PrsNo=457&NewsKeyWd=80x%20sd%20card. [Accessed: 8 Sept. 2007].

[44] "TransTech Wireless Four Camera Security DVR System," Amazon.com, 27 May 2007. [Online]. Available: http://www.amazon.com/TransTech-Wireless-Camera-Security-System/dp/B000R2TAT8/ref=sr_1_3/102-8474263-6808120?ie=UTF8&s=electronics&qid=1187085477&sr=1-3. [Accessed: 14 Aug. 2007].

[45] "Understanding Digital Security Cameras & Security Systems," Peace of Mind Technologies, para. 4. [Online]. Available: http://www.pom-tec.com/CCDSecurityCamera.html. [Accessed: 14 Aug. 2007].

[46] R. Vanzetti, *Practical Applications of Infrared Techniques: A New Tool in a New Dimension For Problem Solving.* Canada: John Wiley and Sons, Inc, pp. 30,41–42,42–43.

[47] "Video resolution and TVL," Indigo Vision Limited. 28 November 2006. [Online]. Available: http://www.indigovision.com/site/modules/White_Papers/IC-COD-REP019%20Video%20Resolution%20and%20TVL.pdf. [Accessed: 17 Aug. 2007].

[48] "Volatile memory," Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Volatile_memory. [Accessed: 17 Aug. 2007].

[49] "VS6724," ST, 8 Jan. 2007. [Online]. Available: http://www.st.com/stonline/products/literature/bd/13026.pdf. [Accessed: 4 Sept. 2007].

[50] E. W. Weisstein, "Curie Temperature," Wolfram Research, 2007. [Online]. Available: http://scienceworld.wolfram.com/physics/CurieTemperature.html. [Accessed: 13 Aug. 2007].

[51] "Wireless 4 Camera IR Weatherproof System," 123 Security Products, para. 1. [Online]. Available: http://www.123securityproducts.com/wi4cairwesyw.html. [Accessed: 14 Aug. 2007].

[52] "Yale HSA 3500 Wirefree Communicating Alarm Kit," LocksOnline, para. 4. [Online]. Available: http://www.locksonline.co.uk/acatalog/Yale_HSA_3500.html. [Accessed: 13 Aug. 2007].

[53] "Welcome to Optical Division of Marshall Electronics," Marshall Electronics, Inc. [Online]. Available: http://www.mars-cam.com/optical.html. [Accessed: 19 Sept. 2007].

A Bill of Materials

BILL OF MATERIALS

University: Worcester Polytechnic Institute, Worcester - MA, US

Sponsor: University of Limerick, Castletroy, Limerick, Ireland

Students: Matthew Conway whitev36@gmail.com

Phong Dam phocada@gmail.com

Janelle L Tavares jtav77@gmail.com

Manager: Mark Southern mark.southern@ul.ie

Seamus Clifford seamus.clifford@ul.ie

Project : Efficient Surveillance System

Advisors: Prof. Vaz

Prof. Brown

Date : 2-Oct-07

Rev : E

Item	Description	Distributor	Part No.	Manufacturer	Module/Interface	Unit Price	Quant.	Sub	Total
1	C328-7640 JPEG Compression VGA Came	Electronics123.com	C328-7640-ST	COMedia Ltd.	Camera	$47.45	3	$142.35	€ 104.45
2	C328-EV232 Evaluation Board for C328 with	Electronics123.com	C328-EV232	COMedia Ltd.	Camera	$49.35	3	$148.05	€ 108.64
3	MSP430-JTAG Programmer	Cool Components	MSP430-JTAG	Olimex Ltd.	Microcontroller	£9.47	2	£18.94	€ 27.98
4	MSP430-P 1611 Prototyping Board	Cool Components	MSP430-P 1611	Olimex Ltd.	Microcontroller	£27.85	1	£27.85	€ 41.14
5	AMN14111 — PANASONIC EW — SENSO	Farnel	6010CA-ND	PANASONIC EW	PIR Sensor	€ 19.96	5	€ 99.80	€ 99.80
6	128Mb Transcend Secure Digital 80x Speed	MemoryC	51KEBK-ND	Transcend	Memory	€ 7.95	4	€ 31.80	€ 31.80
7	Sealed Lead Acid Batteries 6V 8.5AH	Silicon Solar Inc	16525	Silicon Solar Inc	Power Supply	$8.65	3	$25.95	€ 19.04
8	6V Solar Battery Charger	Silicon Solar Inc	SKU2238	Silicon Solar Inc	Power Supply	$39.95	2	$79.90	€ 58.63
9	6V Charge Controller	Silicon Solar Inc	SKU17849	Silicon Solar Inc	Power Supply	$9.95	2	$19.90	€ 14.60
10	Slimline Gender Changer, DB9 Male / Male	L-com	DGB9M	L-com	Camera/MC interface	$4.95	4	$19.80	€ 14.53
11	Do-It-Yourself Kit, DB9 Male / Female	L-com	DIY9MF	L-com	Camera/MC interface	$6.60	4	$26.40	€ 19.37
12	Conn mem secure digital rev SM D	Digikey	HR846CT-ND	Hirose Electric Co Ltd	Memory interface	$4.09	4	$16.36	€ 12.00
13	Proto-Brd 10PIN DISCRETE SMD SIP	Digikey	6010CA-ND	Capital Advanced Tec	Memory interface	$3.08	4	$12.32	€ 9.04
14	RES 51K OHM 1/6W 5% CARBON FILM	Digikey	51KEBK-ND	Yageo Corporation	Memory interface	$0.05	20	$1.04	€ 0.76
15	MSP430F1611 Prototype Board, RS232	Micro Controller Pro	MSP-P1611	Olimex Ltd.	Microcontroller	$45.95	3	$137.85	€ 101.15
16	SD Expansion Board	Crownhill Associate	C328-EV232	GHI Electronics	Memory interface	£9.04	3	£27.12	€ 40.06
17	PQ 18 — FISCHER ELEKTRONIK — SOC	Farnel	4695422	FISCHER ELEKTRON	Microcontroller	€ 0.77	10	€ 7.70	€ 7.70
18	X8M000000L124 — AEL CRYSTALS — CR	Farnel	9509526	AEL CRYSTALS	Microcontroller	€ 1.10	10	€ 11.00	€ 11.00
19	06035A330JAT2A — AVX — CAPACITOR,	Farnel	498555	AVX	Microcontroller	€ 0.05	20	€ 0.94	€ 0.94
20	03243000 — BOPLA — ENCLOSURE ABS	Farnel	1217518	BOPLA	Case	€ 34.05	2	€ 68.10	€ 68.10
21	BT7518 — B-TECH — MOUNT, VESA TILT	Farnel	1277694	B-TECH	Case	€ 14.32	1	€ 14.32	€ 14.32

Grand Total : € 805.06

B Prototype Costs

Table 29: Costs for Camera Node Prototype, Bosman *et al.*

Components for Each Node	Price Break Quantities			
	1	10	100	1000
Camera	€33.89	€30.68	€28.54	€24.98
PIR Sensor	€19.96	€17.41	€15.75	€14.26
MSP430	€32.82	€29.71	€26.89	€24.33
SD Card	€ 7.95	€ 7.95	€ 7.95	€ 7.95
Battery	€ 6.18	€ 6.18	€ 6.18	€ 6.18
Solar Panel	€28.54	€28.54	€25.83	€23.38
Charge Controller	€ 7.11	€ 7.11	€ 6.43	€ 7.11
EM260 RCM	€87.20	€78.92	€71.43	€64.65
Total for Each Node	€223.65	€206.49	€188.99	€171.53

Table 30: Costs for Base Station Prototype, Bosman *et al.*

Components for the Base Station	Price Break Quantities			
	1	10	100	1000
MSP430	€32.82	€29.71	€26.89	€24.33
EM260 RCM	€87.20	€78.92	€71.43	€64.65
Total for Base Station	€120.02	€108.63	€98.32	€88.98

C Solar Panel Testing Data

Table 31: Data collected from the solar panel for 12 hours.

Duration (h)	Current (mA)	Conditions
0.5	0	clear
1.6	39	sunny
2.5	74	sunny
3.3	114	sunny
3.5	128	sunny
4.0	148	sunny
4.5	167	partly cloudy
5.0	199	partly cloudy
5.5	242	sunny
6.0	195	partly cloudy
6.5	71	mostly cloudy
7.0	67	mostly cloudy
7.5	47	overcast
8.0	171	sunny
8.5	144	sunny
9.0	26	partly cloudy
9.5	33	partly cloudy
10.0	113	sunny
10.5	103	sunny
11.0	58	sunny
11.5	31	sunny
12.0	10	clear
12.5	1	clear

Table 32: Data collected from the solar panel for 1 hour.

Duration (min)	Current (mA)
1	361
2	390
3	300
4	264
5	143
6	277
7	320
8	300
9	327
10	300
11	305
12	306
13	219
14	280
15	298
16	289
17	300
18	321
19	319
20	335
21	98
22	157
23	278
24	308
25	333
26	281
27	293
28	293
29	145
30	132

Duration (min)	Current (mA)
31	127
32	145
33	139
34	131
35	116
36	102
37	77
38	78
39	62
40	62
41	70
42	81
43	86
44	77
45	68
46	65
47	72
48	79
49	90
50	90
51	87
52	78
53	77
54	74
55	72
56	66
57	65
58	56
59	73
60	84

D Pictures of different resolutions

Figure 58: 160x128 resolution picture.

Figure 59: 320x240 resolution picture.

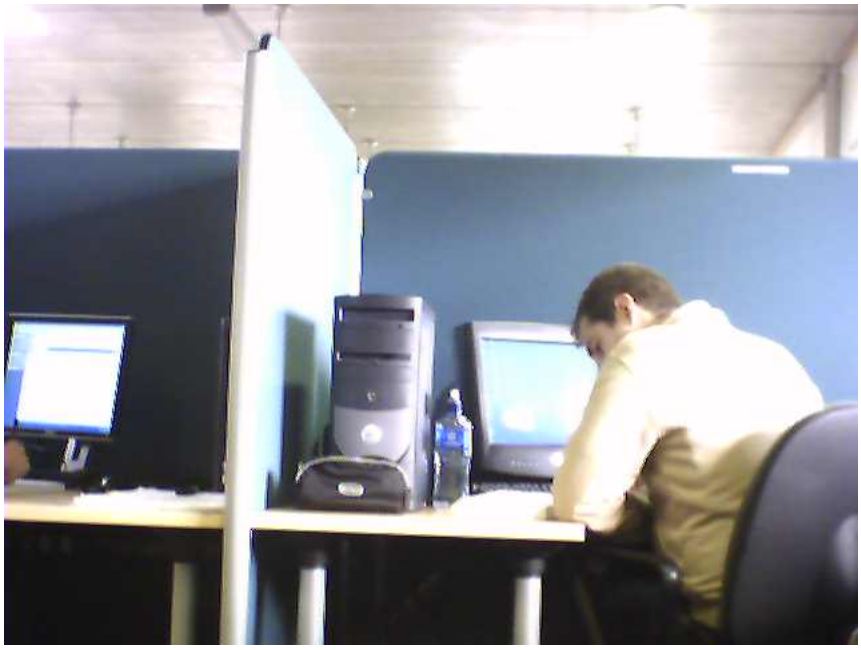
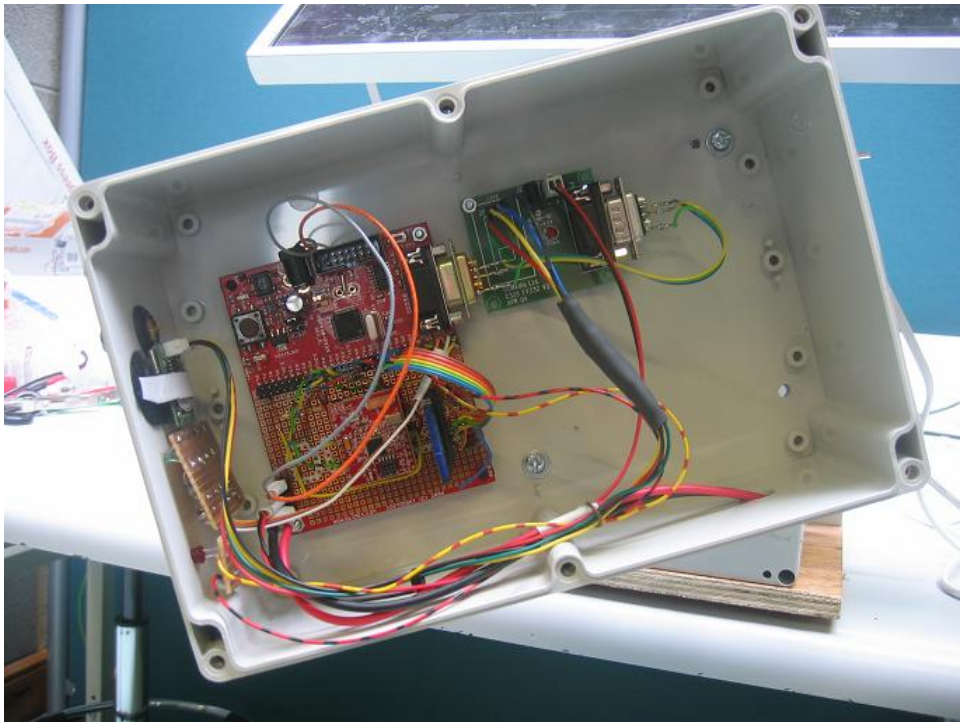


Figure 60: 640x480 resolution picture.

E Picture confirming integrity of JPEG data

F Pictures of complete prototype



G Camera node code

```
/************************************************************
* Project:      EFFICIENT SURVEILLANCE SYSTEM
*
* Univeristy:   WORCESTER POLYTECHNIC INSTITUTE
*               WORCESTER , MA, U.S.A
*
* Advisors:     Prof. Vaz (vaz@wpi.edu)
*               Prof. Brown (drb@wpi.edu)
*
* Students:     Phong Dam (phocada@wpi.edu)
*                     -  Contact for questions regarding codes
*               Matthew Conway (whitey36@wpi.edu)
*               Janelle L. Tavares (jtav77@wpi.edu)
*
* Date:         October 2nd, 2007
*-----------------------------------------------------------
*
* Sponsor:      UNIVERSITY OF LIMERICK
*               CASTLETROY, LIMERICK, IRELAND
* Managers:     Mark Southern (mark.southern@ul.ie)
*               Seamus Clifford (seamus.clifford@ul.ie)
*
*-----------------------------------------------------------
* DESCRIPTION:
*
* This program works with
*      - MSP430F1611
*      - Camera module C328-7640  (communicate with MSP via UART0)
*      - Transcend SD card        (communicate with MSP via SPI1)
*
* The MSP430 is initially set in LPM1 sleep mode. As soon as
* interrupt occurs at P2.0, the MSP will be waken up and start
* sending commands to request the camera to take snapshots.
* The MSP will then request each individual image data packet
* from the camera and transfer it to the SD memory card. The
* user can specify the total number of pictures to be taken
* per interrupt event.
*
```

```
* This program also involves the EM260 Zigbee module which
* communicates with the MSP via SPI0. This module will transmit
* and receive wirelessly the image data store in the SD card.
* However, this module was not within the scope of our project.
*
*****************************************************************/

#include "msp430x16x.h"
#include <math.h>
#include "mmc.h"      // SD card header file


/***********************************
User can specify the number of pictures
to be taken per sensor trigger.
***********************************/
const unsigned short int num_pic_taken = 2;


/********* Image resolution *******
User can set the value of this variable
to obtain desired resolution images.
 - 0x03 : 160 x 128
 - 0x05 : 320 x 240
 - 0x07 : 640 x 480
*********************************/
const unsigned char resolution = 0x07;

unsigned short int timeflag_a = 0;    // interrupt flag for timer A
unsigned short int timeflag_b = 0;    // interrupt flag for timer B
unsigned short int tsec = 0;   // 1/10 second

void runtimera(void);       // start running timer A
void stoptimera(void);      // stop running timber A
void runtimerb(void);       // start running timer B
void stoptimerb(void);      // stop running timber B

/******** System functions ********/
void init_sys(void);    // initialize system: configure clock, pins..
```

```
void reset_sys(void);   // reset all the variables and arrays
void init_uart(void);   // configure UART0
void init_SD(void);     // initialize SD

void wait(void);        // wait 1/10 sec
void wait_a(void);      // wait 1/100 sec
void wait_a_sec(unsigned short int s); // s = 10 ---> wait 0.1 sec
void waitsec(unsigned short int s);  // wait user-specified duration
                                     // s = 10 --> wait 1 sec

/********** LED functions ********/
void LED0n(void);     // turn on the red LED on the development board
void LED0ff(void);    // turn off the red LED
void LED1On(void);    // Green LED 1 --> ON
void LED1Off(void);   // Green LED 1 --> OFF
void LED2On(void);    // Green LED 2 --> ON
void LED2Off(void);   // Green LED 2 --> OFF
void LED3On(void);    // Green LED 3 --> ON
void LED3Off(void);   // Green LED 3 --> OFF
void LED4On(void);    // Green LED 4 --> ON
void LED4Off(void);   // Green LED 4 --> OFF
void blinkLED(unsigned short int i);  // blink the specified LED


/********** Camera commands *******/
char cmdSYNC[6];
char cmdACK[6];
char cmdReset[6];
char cmdPowerOff[6];
char cmdInitial[6];
char cmdSetPackageSize[6];
char cmdSetBaudRate[6];
char cmdSnapShot[6];
char cmdGetPic[6];
char cmdData[6];
char cmdEndPic[6];

/********** Camera constants *******/
const unsigned short int cmdSize = 6;       // each camera command contains 6 bytes
```

```
const unsigned short int cmdACKsize = 12;   // sometimes camera returns 2 commands
const unsigned short int packageSize = 262;   // the size of each packet image data
                                              // the actual image data is 256 bytes

const unsigned short int maxSendSYNC = 60;    // the maxium number of sending SYNC


/*********** Camera functions *******/
void config_camera_cmd(void);      // configure the commands
void send_cmd(char c[]);           // send a specific command to camera
void recv_cmd(void);               // receive an ACK command from camera
void sendSYNC(void);               // send a sequence of SYNC commands to camera
void check(void);                  // check to see if the received command is NACK
void cam_prep(void);               // send 'Initialize' and 'Set package size' commands
void take_pic(void);               // send 'Snap shot' and 'Get picture' commands
void get_image(void);              // get an image from camera
void get_package(unsigned char id);   // get a specific data packet from camera
void cal_numbOfpack(void);            // calculate the number of packets of a taken image
void cal_lastPackSize(void);          // calculate the size of the last packet
void transfer_pack(void);             // transfer the received packets to SD card

void prep_buffer(void);               // prepare the mmc_buffer for writing to SD card
void prep_last_buffer(void);          // prepare the mmc_buffer with the last packet
void clear_mmc_buffer(void);          // reset/clear the mmc_buffer

unsigned short int hex2int(char c1, char c2);   // convert hex to int

unsigned char dummybuffer[262];       // temporarily buffer to store data packets
unsigned short int dummyLen = 0;
unsigned short int dummyRecv = 0;

static unsigned char recvbuffer[262];   // buffer for receiving data from camera
unsigned short int bufFlag = 0;       // this tells whether the packet will be stored
                              // in the first half or second half of the mmc_buffer
                                    // 0 --> first half (256 bytes)
                                    // 1 --> second half (256 bytes)
// variables to work with camera
unsigned short int numberOfpackage;
unsigned short int lastPackDataSize;
```

```
unsigned short int currentLen = 0;
unsigned short int finishSYNCflag = 0;
unsigned short int prepSYNCsuccess = 0;
unsigned short int initialACKflag = 0;
unsigned short int setPackACKflag = 0;
unsigned short int snapACKflag = 0;
unsigned short int getPicACKflag = 0;
unsigned short int dataPackflag = 0;

unsigned short int dataReady = 0; // this signals the MSP430 to start getting packets


/********* SD functions and variable *******/
const unsigned short int timeout = 30;       // duration for initialize SD card
unsigned short int timeoutFlag = 0;   // flag for SD card initialization

unsigned short int block = 0;  // keeping track of which block of SD card is written
                               // Warning: reset this variable will force SD card to write
                               // at first block of address and the old data will be lost
extern char card_state;
char card_state = 0;            // card state: 0: not found, 1 found (init successfull)

extern char mmc_buffer[512];    // buffer for holding data prior to transferring to SD card

/************ Info for EM260 Zigbee module ********/
unsigned short int picSize[100];  // this array stores the size of the each taken picture
unsigned short int numOfPic = 0;   // the total number of pictures taken

void enableUART0(void);
void disableUART0(void);
void transmit_picture(void);

/*=========================================================*/

/************* Timer A ISR *****************/
#pragma vector = TIMERA0_VECTOR
__interrupt void Timer_A0(void)
{
    timeflag_a = 1;
```

```
}

/************* Timer B ISR *****************/
#pragma vector = TIMERB0_VECTOR
__interrupt void Timer_B0(void)
{
    timeoutFlag++;
    timeflag_b = 1;
}

/************* Port 2 ISR *****************/
#pragma vector = PORT2_VECTOR
__interrupt void port_2(void)
{
    _BIC_SR_IRQ(LPM1_bits); // Exit LPM1
    P2IFG = 0;    // clear Port 2 interrupt flag
}

/************* UART0 RX ISR *****************/
#pragma vector = UART0RX_VECTOR     // uart0 receive interrupt
__interrupt void usart0_rx (void)
{
     recvbuffer[currentLen++] = RXBUF0;
     if (dataReady == 1)
        dummybuffer[dummyLen++] = recvbuffer[dummyRecv++];
}

/******************** main () ***************************
*  The system will first enter sleep mode. As soon as the
*  sensor triggers, the system will be activated and start
*  taking and storing 2 pictures onto SD card. It will
*  then go back to sleep mode. The routine repeats.
*********************************************************/

void main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  _BIS_SR(GIE);  // Global interrupt enable
```

```
  init_sys();
  enableUART0();
  init_uart();
  config_camera_cmd();
  LEDOn();
  init_SD();      // initialize SD card
  waitsec(200);   // wait for sensor stablize

  while (1)
  {
    unsigned short int pic_counter = 0; // keep track of how many pictures taken
    LEDOff();
    reset_sys();
    _BIS_SR(LPM1); // LPM1 sleep mode enabled: MCLK and DCO are disabled
                   // SMCLK and ACLK are active

    blinkLED(0);    //  indicate system is activated by sensor
    cam_prep();     // initialize camera by sending multiple SYNC commands
    blinkLED(1);    //  indicate camera is succesfully initialized


    while (pic_counter < num_pic_taken)
    {
      take_pic();     // tell camera to take snapshot
      get_image();    // get image data from camera

      reset_sys();
      pic_counter++;
      blinkLED(3);        // done taking a picture
      waitsec(3);
    }
    send_cmd(cmdReset);
    send_cmd(cmdPowerOff);    // turn off camera
    blinkLED(4);      // done taking user-specified number of pictures

    // disableUART0();
    // transmit_picture();    // after transmitting, SPI0 must be disabled
  }
}
```

```
/****************** init_sys() ************************/
void init_sys(void)
{
  volatile unsigned char dco_cnt = 255;
  BCSCTL1 &= ~XT2OFF;                              // XT2on
  do                                               // wait for MCLK from quartz
  {
    IFG1 &= ~OFIFG;                                // Clear OSCFault flag
    for (dco_cnt = 0xff; dco_cnt > 0; dco_cnt--); // Time for flag to set
  }
  while ((IFG1 & OFIFG) != 0);
  // OSCFault flag still set?
  BCSCTL2 |= SELM_2 + SELS + DIVS_1;
  // MCLK = XT2  SMCLK= MCLK / 2 = 4MHz

  P6DIR |= BIT0 + BIT4 + BIT5 + BIT6 + BIT7;
  // Set P6.0,  P.4 - 7, to output direction

  P6SEL &= ~(BIT0 + BIT4 + BIT5 + BIT6 + BIT7);
  // Make P6.0, P.4 - 7, I/O option

  P2DIR &= ~BIT0;     // Set P2.0 to input direction
  P2SEL &= ~BIT0;     // Make P2.0 I/O option
  P2IES = 0x00;       // Trigger mode: Low to high transition
  P2IE = 0x01;        // Enable Port 2 interrupt
  LED1Off();
  LED2Off();
  LED3Off();
  LED4Off();
}

/****************** init_SD () ************************/
void init_SD (void)
{
  unsigned short int success = 0;

  while (!success)
```

```
  {
    runtimerb();
    while (timeoutFlag < timeout)
    {
      if (initMMC() == MMC_SUCCESS)          // card found
      {
        card_state |= 1;
        success = 1;
      }
      else
      {
        //Error card not detected or rejected during writing
        card_state = 0;                              // no card
      }
    }
    stoptimerb();
    timeoutFlag = 0;
    if (card_state == 0)
      for (int i = 0; i < 3; i++) blinkLED(0);
  }
}

/****************** runtimera() ************************/
void runtimera(void)
{

  TACTL = TASSEL_2 + CNTL_0 + MC_1 + ID_3; // SMCLK, 16bit, up mode, 8 divider
  TACCR0 = 0X1388;  // 5000 SMCLK tics = 0.01 second
  TACCTL0 = CCIE;   // TACCR0 interrupt enabled

  /*
  TACTL = TASSEL_1 + CNTL_0 + MC_1 + ID_0; // ACLK, 16bit, up mode, 1 divider
  TACCR0 = 0x8000;  // 32768 ACLK tics = 1 second
  TACCTL0 = CCIE;   // TACCR0 interrupt enabled
  */
}

/****************** stoptimera() ************************/
void stoptimera(void)
```

```
{
  TACTL = MC_0; // stop timer
  TACCTL0 &= ~CCIE;  // TACCR0 interrupt disabled
}

/****************** runtimerb() ************************/
void runtimerb(void)
{
  TBCTL = TBSSEL_2 + CNTL_0 + MC_1 + ID_3; // SMCLK, 16bit, up mode, 8 divider
  TBCCR0 = 0xC350;  // 50000 SMCLK tics = 0.1 second
  TBCCTL0 = CCIE;   // TBCCR0 interrupt enabled
}

/****************** stoptimerb() ************************/
void stoptimerb(void)
{
  TBCTL = MC_0; // stop timer
  TBCCTL0 &= ~CCIE;  // TBCCR0 interrupt disabled
}

/****************** init_uart() ************************/
void init_uart(void)
{
    P3SEL |= 0x30;            // P3.4 , P3.5 = USART0 option select
    // ME1 |= UTXE0 + URXE0;     // Enable USART0 TXD/RXD
    UCTL0 |= CHAR;            // 8-bit character , NO parity, 1 stop bit
    UTCTL0 |= SSEL1;          // UCLK = SMCLK
    UBR00 = 0x68;             // 4Mhz/38400 ~ 104
    UBR10 = 0x00;             //
    UMCTL0 = 0x4;             // modulation
    UCTL0 &= ~SWRST;          // Initialize USART state machine
    IE1 |= URXIE0;            // Enable USART0 RX interrupt

}

/****************** enableUART0() ********************/
void enableUART0(void)
{
    ME1 |= UTXE0 + URXE0;     // Enable USART0 TXD/RXD
```

```
}
/***************** enableUART0() *********************/
void disableUART0(void)
{
    ME1 &= ~(UTXE0 + URXE0);     // Disable USART0 TXD/RXD
}
/******************* LED0n() ************************/
void LED0n(void)
{
    P6OUT &= ~BIT0; // P6.0 output = 0 (LED off)
}

/******************* LED0ff() ************************/
void LED0ff(void)
{
    P6OUT |= BIT0;  // P6.0 output = 1 (LED off)
}

/******************* LED1On() ************************/
void LED1On(void)
{
    P6OUT &= ~BIT6; // P6.6 output = 0 (LED off)
}

/******************* LED1Off() ************************/
void LED1Off(void)
{
    P6OUT |= BIT6;  // P6.6 output = 1 (LED off)
}
/******************* LED2On() ************************/
void LED2On(void)
{
    P6OUT &= ~BIT7; // P6.7 output = 0 (LED off)
}

/******************* LED2Off() ************************/
void LED2Off(void)
{
    P6OUT |= BIT7;  // P6.7 output = 1 (LED off)
```

```
}
/****************** LED3On() ***********************/
void LED3On(void)
{
    P6OUT &= ~BIT4; // P6.4 output = 0 (LED off)
}

/****************** LED3Off() **********************/
void LED3Off(void)
{
    P6OUT |= BIT4;  // P6.4 output = 1 (LED off)
}

/****************** LED4On() ***********************/
void LED4On(void)
{
    P6OUT &= ~BIT5; // P6.5 output = 0 (LED off)
}

/****************** LED4Off() **********************/
void LED4Off(void)
{
    P6OUT |= BIT5;  // P6.5 output = 1 (LED off)
}

/****************** wait_a() ********************/
void wait_a(void)
{
  runtimera();
  while (!timeflag_a);
  timeflag_a = 0;
  stoptimera();
}

/****************** wait() ********************/
void wait(void)
{
  runtimerb();
  while (!timeflag_b);
```

```
  timeflag_b = 0;
  stoptimerb();
}
/****************** wait_a_sec () ********************/
void wait_a_sec (unsigned short int s)
{
  for (int i = 0; i < s; i++)
    wait_a();
}
/****************** waitsec () ********************/
void waitsec (unsigned short int s)
{
  for (int i = 0; i < s; i++)
    wait();
}

/****************** config_camera_cmd () *********************

  * 2nd element of an array is the command's ID.
  * 3rd, 4th, 5th, and 6th elements of an array are
            Parameter 1, 2, 3, and 4 respectively.

*****************************************************************/
void config_camera_cmd(void)
{

  cmdSYNC[0] = 0xAA;
  cmdSYNC[1] = 0x0D;
  cmdSYNC[2] = 0x00;
  cmdSYNC[3] = 0x00;
  cmdSYNC[4] = 0x00;
  cmdSYNC[5] = 0x00;

  cmdACK[0] = 0xAA;
  cmdACK[1] = 0x0E;
  cmdACK[2] = 0x0D;
  cmdACK[3] = 0x00;
  cmdACK[4] = 0x00;
  cmdACK[5] = 0x00;
```

```
cmdReset[0] = 0xAA;
cmdReset[1] = 0x08;
cmdReset[2] = 0x00;
cmdReset[3] = 0x00;
cmdReset[4] = 0x00;
cmdReset[5] = 0xFF;

cmdPowerOff[0] = 0xAA;
cmdPowerOff[1] = 0x09;
cmdPowerOff[2] = 0x00;
cmdPowerOff[3] = 0x00;
cmdPowerOff[4] = 0x00;
cmdPowerOff[5] = 0x00;

cmdInitial[0] = 0xAA;
cmdInitial[1] = 0x01;
cmdInitial[2] = 0x00;
cmdInitial[3] = 0x07;   // JPEG file
cmdInitial[4] = 0x00;
cmdInitial[5] = resolution;


// package size is 262 bytes = 0x0106h
cmdSetPackageSize[0] = 0xAA;
cmdSetPackageSize[1] = 0x06;
cmdSetPackageSize[2] = 0x08;
cmdSetPackageSize[3] = 0x06;    // Low Byte size
cmdSetPackageSize[4] = 0x01;    // High Byte size
cmdSetPackageSize[5] = 0x00;

// the baud rate = 38400 bps
cmdSetBaudRate[0] = 0xAA;
cmdSetBaudRate[1] = 0x07;
cmdSetBaudRate[2] = 0x2F;
cmdSetBaudRate[3] = 0x01;
cmdSetBaudRate[4] = 0x00;
cmdSetBaudRate[5] = 0x00;
```

```
  cmdSnapShot[0] = 0xAA;
  cmdSnapShot[1] = 0x05;
  cmdSnapShot[2] = 0x00;
  cmdSnapShot[3] = 0x00;
  cmdSnapShot[4] = 0x00;
  cmdSnapShot[5] = 0x00;

  cmdGetPic[0] = 0xAA;
  cmdGetPic[1] = 0x04;
  cmdGetPic[2] = 0x01;  // snap mode
  cmdGetPic[3] = 0x00;
  cmdGetPic[4] = 0x00;
  cmdGetPic[5] = 0x00;

  cmdEndPic[0] = 0xAA;
  cmdEndPic[1] = 0x0E;
  cmdEndPic[2] = 0x00;
  cmdEndPic[3] = 0x00;
  cmdEndPic[4] = 0xF0;
  cmdEndPic[5] = 0xF0;
}

/****************** send_cmd() ***********************/
void send_cmd(char c[])
{
  // send a command
  for(int i = 0; i < cmdSize; i++)
  {
     while (!(IFG1 & UTXIFG0));     // USART0 TX buffer ready?
     TXBUF0 = c[i];
  }
}

/********************** sendSYNC () ********************/
void sendSYNC(void)
{
  // This loop will keep sending SYNC command to the camera
  // until the camera sends back ACK and SYNC command.
    while (!finishSYNCflag)
```

```
    {
      int i = 0;
      i++;
      if (i > maxSendSYNC)
      {
       send_cmd(cmdReset);
       send_cmd(cmdPowerOff);
      }
      send_cmd(cmdSYNC);
      wait_a_sec(5);
      if (currentLen == cmdACKsize) // receive complete message
      {
        currentLen = 0;
        finishSYNCflag = 1;
      }
    }
}

/********************** cam_prep () *********************/
void cam_prep (void)
{

   while(!prepSYNCsuccess)
   {
     sendSYNC();
     wait_a_sec(5);
     // The system sends back ACK command to camera
     send_cmd(cmdACK);
     wait();
     check();
   }

    // send initial command and wait for ACK
    send_cmd(cmdInitial);
    while (!initialACKflag)
    {
       if (currentLen == cmdSize)
       {
         currentLen = 0;
```

```
          initialACKflag = 1;
        }
    }
    wait_a_sec(5);

    // send set package size command and wait for ACK
    send_cmd(cmdSetPackageSize);
    while (!setPackACKflag)
    {
        if (currentLen == cmdSize)
        {
          currentLen = 0;
          setPackACKflag = 1;
        }
    }
     wait_a_sec(5);
}

/********************** take_pic () *********************/
void take_pic(void)
{
    // send snapshot command and wait for ACK
    send_cmd(cmdSnapShot);
    while (!snapACKflag)
    {
        if (currentLen == cmdSize)
        {
          currentLen = 0;
          snapACKflag = 1;
        }
    }
    wait_a_sec(5);

    // send get picture command and wait for ACK
    send_cmd(cmdGetPic);
    while (!getPicACKflag)
    {
        if (currentLen == cmdACKsize)
        {
```

```
          currentLen = 0;
          getPicACKflag = 1;
        }
    }
    cal_numbOfpack();
}

/********************** get_image () ********************/
void get_image(void)
{

  dataReady = 1;   // start receiving data and put them into dummybuffer[]
  currentLen = 0;

  unsigned short int packID = 0x00;
  // get each package from camera and dump it onto SD
  while (packID < numberOfpackage)
  {
    get_package(packID);
    wait();
    if (packID != numberOfpackage - 1)  // if packID is not the last packet
    {
      prep_buffer();
    }
    else
    {
      prep_last_buffer();
    }
    packID++;
    currentLen = 0;
    dummyLen = 0;
    dummyRecv = 0;
  }
  wait();
  send_cmd(cmdEndPic);
  dataReady = 0;    // stop storing into dummybuffer[]
}

/********************** get_package () ********************/
```

```
void get_package(unsigned char id)
{
  char cmd[6];
  cmd[0] = 0xAA;
  cmd[1] = 0x0E;
  cmd[2] = 0X0A;
  cmd[3] = 0X00;
  cmd[4] = id;
  cmd[5] = 0x00;

  // send a command
  for(int i = 0; i < cmdSize; i++)
  {
     while (!(IFG1 & UTXIFG0));     // USART0 TX buffer ready?
     TXBUF0 = cmd[i];
  }
}

/*************** prep_buffer() **************/
void prep_buffer(void)
{
  if (bufFlag == 0)
  {
    clear_mmc_buffer();
    for (int i = 4; i<260 ; i++)
      mmc_buffer[i-4] = dummybuffer[i];
    bufFlag = 1;
  }
  else
  {
    for (int i = 4 ; i < 260; i++)
      mmc_buffer[i+252] = dummybuffer[i];
    transfer_pack();
    blinkLED(2);
    bufFlag = 0;
  }
}

/*************** prep_last_buffer() **************/
```

```
void prep_last_buffer(void)
{
  cal_lastPackSize();     // get last package size
  if (bufFlag == 0)
  {
    clear_mmc_buffer();
    for (int i = 4; i < (lastPackDataSize + 4); i++)
      mmc_buffer[i-4] = dummybuffer[i];
    transfer_pack();
    blinkLED(2);
  }
  else
  {
    for (int i = 4 ; i < (lastPackDataSize + 4); i++)
      mmc_buffer[i+252] = dummybuffer[i];
    transfer_pack();
    blinkLED(2);
  }
}
/*************** transfer_pack() **************/
void transfer_pack(void)
{
    wait_a();
    mmcWriteBlock(512*(block++));    // write to SD card
    wait_a();
}

/*************** clear_mmc_buffer() **************/
void clear_mmc_buffer(void)
{
  for (int i = 0 ; i < 512 ; i++)
    mmc_buffer[i] = 0x00;
}

/*************** check() ******************/
void check(void)
{
  if (recvbuffer[1] == 0x0F || recvbuffer[7] == 0x0F)
  {
```

```
    send_cmd(cmdReset);
    send_cmd(cmdPowerOff);
  }
  else prepSYNCsuccess = 1;
}

/*************** cal_numbOfpack() ******************/
void cal_numbOfpack(void)
{
  unsigned short int totalBytes  = 0 ;
  totalBytes = hex2int (recvbuffer[9], recvbuffer[10]);
  picSize[numOfPic++] = totalBytes;
  numberOfpackage = (totalBytes / (packageSize - 6)) + 1;
}

/*************** cal_lastPackSize() *************/
void cal_lastPackSize(void)
{
  lastPackDataSize = hex2int (recvbuffer[2], recvbuffer[3]);
}

/*************** hex2int() **********************/
unsigned short int hex2int(char c1, char c2)
{

  unsigned short int i1,i2,i3,i4;
//  unsigned short int i5,i6;
  unsigned short int total  = 0 ;

  i1 = (c1 & 0x0F);
  i2 = ((c1 & 0xF0) >> 4) * 16;
  i3 = (c2 & 0X0F) * 256;
  i4 = ((c2 & 0xF0) >> 4) * 4096;

//  i5 = (c3 & 0X0F) * 65536;
//  i6 = ((c3 & 0xF0) >> 4) * 1048576;
//  totalBytes = i1+i2+i3+i4+i5+i6;
  total = i1+i2+i3+i4;
  return total;
```

```
}
/*************** blinkLED() ******************/
void blinkLED(unsigned short int i)
{
  switch (i)
  {
  case 0:
    LED0n();
    waitsec(2);
    LED0ff();
    waitsec(2);
    break;
  case 1:
    LED1On();
    waitsec(2);
    LED1Off();
    waitsec(2);
    break;
  case 2:
    LED2On();
    waitsec(2);
    LED2Off();
    waitsec(2);
    break;
  case 3:
    LED3On();
    waitsec(2);
    LED3Off();
    waitsec(2);
    break;
  case 4:
    LED4On();
    waitsec(2);
    LED4Off();
    waitsec(2);
    break;
  default:
    break;
  }
```

```
}

/*************** reset_sys() ******************/
void reset_sys(void)
{

  timeflag_a = 0;
  timeflag_b = 0;
  tsec = 0;
  dataReady = 0;
  bufFlag = 0;
  timeoutFlag = 0;
  dummyLen = 0;
  dummyRecv = 0;

  finishSYNCflag = 0;
  prepSYNCsuccess = 0;
  initialACKflag = 0;
  setPackACKflag = 0;
  snapACKflag = 0;
  getPicACKflag = 0;
  dataPackflag = 0;
  currentLen = 0;
  numberOfpackage = 0;
  lastPackDataSize = 0;

  for (int i = 0; i < 262; i++) {
   dummybuffer[i] = 0x00;
   recvbuffer[i] = 0x00;
  }

  for (int j = 0; j < 512; j++)
    mmc_buffer[j] = 0x00;
}
```

```
/*
  mmc.h: Dekcarations for Communikation with the MMC
  (see mmc.c) in unprotected spi mode.

Pin configuration at MSP430F149:
---------------------------------
  MC                    MC Pin          MMC              MMC Pin
  P5.4                  48              ChipSelect       1
  P5.1 / SlaveInMasterOut 45            DataIn           2
  .                                     GND              3 (0 V)
  .                                     VDD              4 (3.3 V)
  P5.3 UCLK1 / SlaveCLocK 47            Clock            5
  .                                     GND              6 (0 V)
  P5.2 / SlaveOutMasterIn 46            DataOut          7
  P5.4                                  CardDetect with pullup
-----------------------------------------------------------------------

 Revisions
 Date           Author                                  Revision
 11. May 2003           Rolf Freitag                            0.02
(2004: corrected MC pin numbers (switched only 45, 46))
*/
/*
 * ------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 44):
 * Rolf Freitag (webmaster at true-random.com) wrote this file.
 * As long as you retain this notice you can do whatever
 * the LGPL (Lesser GNU public License) allows with this stuff.
 * If you think this stuff is worth it, you can send me money via
 * paypal or if we met some day you can buy me a beer in return.
 * ------------------------------------------------------------------
 */
#ifndef _MMCLIB_H
#define _MMCLIB_H

#ifndef TXEPT                              // transmitter-empty flag
#define TEXPT 0x01
#endif
```

```
// macro defines
#define HIGH(a) ((a>>8)&0xFF)                    // high byte from word
#define LOW(a) (a&0xFF)                          // low byte from word

#define CS_LOW()  P5OUT &= ~0x01                 // Card Select
#define CS_HIGH() P5OUT |= 0x01                  // Card Deselect
#define SPI_RXC (IFG2 & URXIFG1)
#define SPI_TXC (IFG2 & UTXIFG1)

#define SPI_RX_COMPLETE (IFG2 & URXIFG1)
#define SPI_TX_READY (IFG2 & UTXIFG1)

#define DUMMY 0xff

// Tokens (nessisary because at nop/idle (and CS active)
only 0xff is on the data/command line)
#define MMC_START_DATA_BLOCK_TOKEN    0xfe
// Data token start byte, Start Single Block Read
#define MMC_START_DATA_MULTIPLE_BLOCK_READ  0xfe
// Data token start byte, Start Multiple Block Read
#define MMC_START_DATA_BLOCK_WRITE    0xfe
// Data token start byte, Start Single Block Write
#define MMC_START_DATA_MULTIPLE_BLOCK_WRITE 0xfc
// Data token start byte, Start Multiple Block Write
#define MMC_STOP_DATA_MULTIPLE_BLOCK_WRITE  0xfd
// Data toke stop byte, Stop Multiple Block Write

// an affirmative R1 response (no errors)
#define MMC_R1_RESPONSE       0x00

// this variable will be used to track the current block length
// this allows the block length to be set only when needed
// unsigned long _BlockLength = 0;

// error/success codes
#define MMC_SUCCESS           0x00
#define MMC_BLOCK_SET_ERROR   0x01
#define MMC_RESPONSE_ERROR    0x02
#define MMC_DATA_TOKEN_ERROR  0x03
```

```
#define MMC_INIT_ERROR        0x04
#define MMC_CRC_ERROR         0x10
#define MMC_WRITE_ERROR       0x11
#define MMC_OTHER_ERROR       0x12
#define MMC_TIMEOUT_ERROR     0xFF

// commands: first bit 0 (start bit), second 1 (transmission bit);
//CMD-number + 0ffsett 0x40
#define MMC_GO_IDLE_STATE   0x40                  //CMD0
#define MMC_SEND_OP_COND  0x41                    //CMD1
#define MMC_READ_CSD    0x49                      //CMD9
#define MMC_SEND_CID    0x4a                      //CMD10
#define MMC_STOP_TRANSMISSION   0x4c              //CMD12
#define MMC_SEND_STATUS   0x4d                    //CMD13
#define MMC_SET_BLOCKLEN  0x50                    //CMD16
//Set block length for next read/write
#define MMC_READ_SINGLE_BLOCK   0x51              //CMD17
//Read block from memory
#define MMC_READ_MULTIPLE_BLOCK 0x52              //CMD18
#define MMC_CMD_WRITEBLOCK  0x54                  //CMD20
//Write block to memory
#define MMC_WRITE_BLOCK   0x58                    //CMD25
#define MMC_WRITE_MULTIPLE_BLOCK 0x59             //CMD??
#define MMC_WRITE_CSD     0x5b                    //CMD27 PROGRAM_CSD
#define MMC_SET_WRITE_PROT  0x5c                  //CMD28
#define MMC_CLR_WRITE_PROT  0x5d                  //CMD29
#define MMC_SEND_WRITE_PROT   0x5e                //CMD30
#define MMC_TAG_SECTOR_START  0x60                //CMD32
#define MMC_TAG_SECTOR_END  0x61                  //CMD33
#define MMC_UNTAG_SECTOR  0x62                    //CMD34
#define MMC_TAG_EREASE_GROUP_START 0x63           //CMD35
#define MMC_TAG_EREASE_GROUP_END 0x64             //CMD36
#define MMC_UNTAG_EREASE_GROUP  0x65              //CMD37
#define MMC_EREASE    0x66                        //CMD38
#define MMC_READ_OCR    0x67                      //CMD39
#define MMC_CRC_ON_OFF    0x68                    //CMD40

//TI added sub function for top two spi_xxx
```

```
// mmc init
char initMMC (void);
// send command to MMC
void mmcSendCmd (const char cmd, unsigned long data, const char crc);
// set MMC block length of count=2^n Byte
char mmcSetBlockLength (const unsigned long);
// read a size Byte big block beginning at the address.
char mmcReadBlock(const unsigned long address, const unsigned long count);
// write a 512 Byte big block beginning at the (aligned) adress
char mmcWriteBlock (const unsigned long address);
// Register arg1 der Laenge arg2 auslesen (into the buffer)
char mmcReadRegister(const char, const unsigned char);
#endif                                          /* _MMCLIB_H */
```

```
// Rolf Freitag 5/2003
/*
 * ------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 44):
 * Rolf Freitag (webmaster at true-random.com) wrote this file.
 * As long as you retain this notice you can do whatever
 * the LGPL (Lesser GNU public License) allows with this stuff.
 * If you think this stuff is worth it, you can send me money via
 * paypal or if we met some day you can buy me a beer in return.
 * ------------------------------------------------------------
 */

// MMC Lib
#ifndef _MMCLIB_C
#define _MMCLIB_C
//-------------------------------------------------------------
#include "mmc.h"

#include  "MSP430x16x.H"
#include  "math.h"
#include  "string.h"

char mmcGetResponse(void);
char mmcGetXXResponse(const char resp);
char mmcCheckBusy(void);

void initSPI (void);
unsigned char spiSendByte(const unsigned char data);

char mmc_buffer[512] =
// Buffer for mmc i/o for data and registers
{
  0
};

extern char card_state;
// 0 for no card found, 1 for card found (init successfull)

//-------------------------------------------------------------------
```

```
// setup usart1 in spi mode
void initSPI (void)
{
  ME2 |= USPIE1;                                    // Enable USART1 SPI mode
//  UTCTL1 = CKPH | SSEL1 | SSEL0 | STC;
// SMCLK, 3-pin mode, clock idle low, data valid on rising edge, UCLK delayed
  UTCTL1 = CKPH | SSEL1 | STC;
  // SMCLK, 3-pin mode, clock idle low, data valid on rising edge, UCLK delayed
//  UBR01 = 0x02;
// 0x02: UCLK/2 (4 MHz), works also with 3 and 4
  UBR01 = 0X0A;                                     // 400KHz for initialization
  UBR11 = 0x00;                                     // -"-
  UMCTL1 = 0x00;                                    // no modulation
  UCTL1 = CHAR | SYNC | MM;                         // 8-bit SPI Master **SWRST**
  P5SEL |= 0x0E;                                    // P5.1-3 SPI option select
  P5DIR |= 0x01;                                    // P5.0 output direction
//  P5OUT = 0xff;
  P5OUT |= 0x01;
  U1CTL &= ~SWRST; // SPI enable
  while (!(IFG2 & UTXIFG1));
  // USART1 TX buffer ready (empty)?
  // debug_printf("init......SPI");
}


// Initialisieren
char initMMC (void)
{

  //raise SS and MOSI for 80 clock cycles
  //SendByte(0xff) 10 times with SS high
  //RAISE SS
  int i;
  char response=0x01;

  // debug_printf("Start iniMMC......");
  initSPI();
  //initialization sequence on PowerUp
```

```
  CS_HIGH();
  for(i=0;i<=9;i++)
    spiSendByte(0xff);
  CS_LOW();
  //Send Command 0 to put MMC in SPI mode
  mmcSendCmd(0x00,0,0x95);
  //Now wait for READY RESPONSE
  if(mmcGetResponse()!=0x01);
  //       debug_printf("no responce");

  while(response==0x01)
  {
    //  debug_printf("Sending Command 1");
    CS_HIGH();
    spiSendByte(0xff);
    CS_LOW();
    mmcSendCmd(0x01,0x00,0xff);
    response=mmcGetResponse();
  }
  CS_HIGH();
  spiSendByte(0xff);
  // debug_printf("MMC INITIALIZED AND SET TO SPI MODE PROPERLY.");
  UBR01 = 0X02;         // 2MHz SPI communication
  return MMC_SUCCESS;
}


// Ti added mmc Get Responce
char mmcGetResponse(void)
{
  //Response comes 1-8bytes after command
  //the first bit will be a 0
  //followed by an error code
  //data will be 0xff until response
  int i=0;

  char response;

  while(i<=64)
```

```
  {
    response=spiSendByte(0xff);
    if(response==0x00)break;
    if(response==0x01)break;
    i++;
  }
  return response;
}


char mmcGetXXResponse(const char resp)
{
  //Response comes 1-8bytes after command
  //the first bit will be a 0
  //followed by an error code
  //data will be 0xff until response
  int i=0;

  char response;

  while(i<=500)
  {
    response=spiSendByte(0xff);
    if(response==resp)break;
    i++;
  }
  return response;
}


char mmcCheckBusy(void)
{
  //Response comes 1-8bytes after command
  //the first bit will be a 0
  //followed by an error code
  //data will be 0xff until response
  int i=0;

  char response;
```

```
  char rvalue;
  while(i<=64)
  {
    response=spiSendByte(0xff);
    response &= 0x1f;
    switch(response)
    {
      case 0x05: rvalue=MMC_SUCCESS;break;
      case 0x0b: return(MMC_CRC_ERROR);
      case 0x0d: return(MMC_WRITE_ERROR);
      default:
        rvalue = MMC_OTHER_ERROR;
        break;
    }
    if(rvalue==MMC_SUCCESS)break;
    i++;
  }
  i=0;
  do
  {
    response=spiSendByte(0xff);
    i++;
  }while(response==0);
  return response;
}


// The card will respond with a standard response token followed by a data
// block suffixed with a 16 bit CRC.

// Ti Modification: long int -> long ; int -> long
char mmcReadBlock(const unsigned long address, const unsigned long count)
{
  unsigned long i = 0;
  char rvalue = MMC_RESPONSE_ERROR;

  // Set the block length to read
  if (mmcSetBlockLength (count) == MMC_SUCCESS)   // block length could be set
  {
```

```
  // SS = LOW (on)
  CS_LOW ();
  // send read command MMC_READ_SINGLE_BLOCK=CMD17
  mmcSendCmd (17,address, 0xFF);
  // Send 8 Clock pulses of delay, check if the
  //MMC acknowledged the read block command
  // it will do this by sending an affirmative response
  // in the R1 format (0x00 is no errors)
  if (mmcGetResponse() == 0x00)
  {
    // now look for the data token to signify the start of
    // the data
    if (mmcGetXXResponse(MMC_START_DATA_BLOCK_TOKEN)
    == MMC_START_DATA_BLOCK_TOKEN)
    {
      // clock the actual data transfer and receive the bytes;
      //spi_read automatically finds the Data Block
      for (i = 0; i < 512; i++)
        mmc_buffer[i] = spiSendByte(0xff);
        // is executed with card inserted

      // get CRC bytes (not really needed by us, but required by MMC)
      spiSendByte(0xff);
      spiSendByte(0xff);
      rvalue = MMC_SUCCESS;
    }
    else
    {
      // the data token was never received
      rvalue = MMC_DATA_TOKEN_ERROR;            // 3
    }
  }
  else
  {
    // the MMC never acknowledge the read command
    rvalue = MMC_RESPONSE_ERROR;                // 2
  }
}
else
```

```
  {
    rvalue = MMC_BLOCK_SET_ERROR;                  // 1
  }
  CS_HIGH ();
  spiSendByte(0xff);
  return rvalue;
}                                                  // mmc_read_block


//-----------------------------------------------------------------------
// Ti Modification: long int -> long
char mmcWriteBlock (const unsigned long address)
{
  unsigned long i = 0;
  char rvalue = MMC_RESPONSE_ERROR;
  // MMC_SUCCESS;
  char c = 0x00;

  // Set the block length to read
  if (mmcSetBlockLength (512) == MMC_SUCCESS)
  // block length could be set
  {
    // SS = LOW (on)
    CS_LOW ();
    // send write command
    mmcSendCmd (24,address, 0xFF);

    // check if the MMC acknowledged the write block command
    // it will do this by sending an affirmative response
    // in the R1 format (0x00 is no errors)
    if (mmcGetXXResponse(MMC_R1_RESPONSE) == MMC_R1_RESPONSE)
    {
      spiSendByte(0xff);
      // send the data token to signify the start of the data
      spiSendByte(0xfe);
      // clock the actual data transfer and transmitt the bytes
      for (i = 0; i < 512; i++)
        spiSendByte(mmc_buffer[i]);
        // mmc_buffer[i];       Test: i & 0xff
```

```
      // put CRC bytes (not really needed by us, but required by MMC)
      spiSendByte(0xff);
      spiSendByte(0xff);
      // read the data response xxx0<status>1 : status 010:
      //Data accected, status 101: Data
      //   rejected due to a crc error, status 110: Data
      //rejected due to a Write error.
      mmcCheckBusy();
    }
    else
    {
      // the MMC never acknowledge the write command
      rvalue = MMC_RESPONSE_ERROR;                // 2
    }
  }
  else
  {
    rvalue = MMC_BLOCK_SET_ERROR;                 // 1
  }
  // give the MMC the required clocks to finish up
  //whatever it needs to do
  //  for (i = 0; i < 9; ++i)
  //    spiSendByte(0xff);

  CS_HIGH ();
  // Send 8 Clock pulses of delay.
  spiSendByte(0xff);
  return rvalue;
}                                                 // mmc_write_block


//-----------------------------------------------------------------------
void mmcSendCmd (const char cmd, unsigned long data, const char crc)
{
  char frame[6];
  char temp;
  int i;

  frame[0]=(cmd|0x40);
```

```
  for(i=3;i>=0;i--)
  {
    temp=(char)(data>>(8*i));
    frame[4-i]=(temp);
  }
  frame[5]=(crc);
  for(i=0;i<6;i++)
    spiSendByte(frame[i]);
}


//--------------- set blocklength 2^n --------------------------
// Ti Modification: long int-> long
char mmcSetBlockLength (const unsigned long blocklength)
{
  char rValue = MMC_TIMEOUT_ERROR;
  char i = 0;

  // SS = LOW (on)
  CS_LOW ();

  // Set the block length to read
  //MMC_SET_BLOCKLEN =CMD16
  mmcSendCmd(16, blocklength, 0xFF);

  // get response from MMC - make sure that its 0x00 (R1 ok response format)
  if(mmcGetResponse()!=0x00);

  CS_HIGH ();

  // Send 8 Clock pulses of delay.
  spiSendByte(0xff);

  return MMC_SUCCESS;
}                                                 // block_length


//TI added substitution routine for spi_read and spi_write
```

```
unsigned char spiSendByte(const unsigned char data)
{
  while ((IFG2&UTXIFG1) ==0);                    // wait while not ready / for RX
  TXBUF1 = data;                                 // write
  while ((IFG2 & URXIFG1)==0);                   // wait for RX buffer (full)
  return (RXBUF1);
}


// Reading the contents of the CSD and CID registers in SPI mode is a simple
// read-block transaction.

char mmcReadRegister (const char cmd_register, const unsigned char length)
{
  char uc = 0;
  char rvalue = MMC_TIMEOUT_ERROR;
  //  char i = 0;

  if (mmcSetBlockLength (length) == MMC_SUCCESS)
  {
    CS_LOW ();
    // CRC not used: 0xff as last byte
    mmcSendCmd(cmd_register, 0x000000, 0xff);

    // wait for response
    // in the R1 format (0x00 is no errors)
    if (mmcGetResponse() == 0x00)
    {
      if (mmcGetXXResponse(0xfe)== 0xfe)
        for (uc = 0; uc < length; uc++)
          mmc_buffer[uc] = spiSendByte(0xff);
      // get CRC bytes (not really needed by us, but required by MMC)
      spiSendByte(0xff);
      spiSendByte(0xff);
    }
    else
      rvalue = MMC_RESPONSE_ERROR;
    // CS = HIGH (off)
    CS_HIGH ();
```

```
    // Send 8 Clock pulses of delay.
    spiSendByte(0xff);
  }
  CS_HIGH ();
  return rvalue;
}                                                   // mmc_read_register


//-----------------------------------------------------------------------
#endif                                              /* _MMCLIB_C */
```

```
/*****************************************************************
* Project:      EFFICIENT SURVEILLANCE SYSTEM
*
* Univeristy:   WORCESTER POLYTECHNIC INSTITUTE, WORCESTER , MA, U.S.A
*
* Advisors:     Prof. Vaz (vaz@wpi.edu)
*               Prof. Brown (drb@wpi.edu)
*
* Students:     Phong Dam (phocada@wpi.edu)
*               -  Contact person for questions regarding this program code
*               Matthew Conway (whitey36@wpi.edu)
*               Janelle L. Tavares (jtav77@wpi.edu)
*
* Date:         October 2nd, 2007
*----------------------------------------------------------------------
*
* Sponsor:      UNIVERSITY OF LIMERICK, CASTLETROY, LIMERICK, IRELAND
* Managers:     Mark Southern (mark.southern@ul.ie)
*               Seamus Clifford (seamus.clifford@ul.ie)
*
*----------------------------------------------------------------------
* Program description:  This program works with
*                           - MSP430F1611
*                           - Desktop computer(communicate with MSP via UART0)
*                           - Transcend SD card(communicate with MSP via SPI1)
*
* The program simply requests the MSP to output the user-defined blocks
*of data from the SD card to the computer.
*
*****************************************************************/

#include "msp430x16x.h"
#include <string.h>
#include <math.h>
#include "mmc.h"

const unsigned short int start_block = 0;
const unsigned short int end_block = 50;
```

```
unsigned short int timeflag_b = 0;
unsigned short int tsec = 0;   // 1/10 second

void runtimerb(void);
void stoptimerb(void);

void LEDOn(void);
void LEDOff(void);
void init_sys(void);
void init_uart(void);
void wait(void);

// SD card
extern char card_state;
extern char mmc_buffer[512];
char card_state = 0;
// card state: 0: not found, 1 found (init successful)
void read_data(void);


/************* Timer B ISR *****************/
#pragma vector = TIMERB0_VECTOR
__interrupt void Timer_B0(void)
{
  tsec++;   // increment 0.1s
  if (tsec == 3)     // 0.3s second passes
  {
    timeflag_b = 1;
    tsec = 0;
  }
}

/******************** main () ***************************/

void main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
```

```
  _BIS_SR(GIE);  // Global interrupt enable
  init_sys();
  init_uart();

  for (int i = 0; i < 512; i++)   // clear buffer
    mmc_buffer[i] = 0x00;

  LEDOn();
  read_data();    // output image data to PC through UART0
  LEDOff();

}

/******************** read_data () ********************/
void read_data(void)
{
    unsigned short int blockSize = 512;
    //init mmc card

    if (initMMC() == MMC_SUCCESS)                 // card found
    {
      card_state |= 1;

      wait();
      for (int i = start_block; i < end_block ; i++)
      {

        mmcReadBlock(blockSize*i,blockSize);
        for (int j = 0 ; j < blockSize ; j++) {
           while (!(IFG1 & UTXIFG0));     // USART0 TX buffer ready?
           TXBUF0 = mmc_buffer[j];
       }
      }
      LEDOn();
    }
    else
    {
      //Error card not detected or rejected during writing
      card_state = 0;                             // no card
```

```
    }

}
/****************** init_sys() ************************/
void init_sys(void)
{
  volatile unsigned char dco_cnt = 255;
  BCSCTL1 &= ~XT2OFF;                          // XT2on
  do                                          // wait for MCLK from quartz
  {
    IFG1 &= ~OFIFG;                                // Clear OSCFault flag
    for (dco_cnt = 0xff; dco_cnt > 0; dco_cnt--); // Time for flag to set
  }
  while ((IFG1 & OFIFG) != 0);                     // OSCFault flag still set?
  BCSCTL2 |= SELM_2 + SELS + DIVS_1;    // MCLK = XT2  SMCLK= MCLK / 2 = 4MHz

  P6DIR |= BIT0;     // Set P6.0 to output direction
  P6SEL &= ~BIT0;    // Make P6.0 I/O option
}

/****************** runtimerb() ************************/
void runtimerb(void)
{
  TBCTL = TBSSEL_2 + CNTL_0 + MC_1 + ID_3; // SMCLK, 16bit, up mode, 8 divider
  TBCCR0 = 0xC350;  // 50000 SMCLK tics = 0.1 second
  TBCCTL0 = CCIE;   // TBCCR0 interrupt enabled
}

/****************** stoptimerb() ************************/
void stoptimerb(void)
{
  TBCTL = MC_0; // stop timer
  TBCCTL0 &= ~CCIE;  // TBCCR0 interrupt disabled
}

/****************** init_uart() ************************/
void init_uart(void)
{
    P3SEL |= 0x30;                     // P3.4 , P3.5 = USART0 option select
```

```
    ME1 |= UTXE0 + URXE0;             // Enable USART0 TXD/RXD
    UCTL0 |= CHAR;                    // 8-bit character , NO parity, 1 stop bit
    UTCTL0 |= SSEL1;                  // UCLK = SMCLK
    UBR00 = 0x68;                     // 4Mhz/38400 ~ 104
    UBR10 = 0x00;                     //
    UMCTL0 = 0x4;                     // modulation
    UCTL0 &= ~SWRST;                  // Initialize USART state machine
    IE1 |= URXIE0;                    // Enable USART0 RX interrupt

}
/******************* LED0n() ************************/
void LED0n(void)
{
    P6OUT &= ~BIT0; // P6.0 output = 0 (LED off)
}

/******************* LED0ff() ***********************/
void LED0ff(void)
{
    P6OUT |= BIT0;  // P6.0 output = 1 (LED off)
}

/******************* wait() *********************/
void wait(void)
{
  runtimerb();
  while (!timeflag_b);
  timeflag_b = 0;
  stoptimerb();
}
```

H Base station code

```
/*
Base Station Code - index.php
Ireland A07 MQP - Efficient Surveillance System
Advisors: Prof. Vaz, Prof. Brown
Students: Matt Conway, Phong Dam, Janelle Tavares*
*author of this code

Summary:
This file contains the code for the main page of
the website.  This webpage periodically checks
its directory(our share of the UL web server)
for new images. If there are new images, they
are uploaded first, indicated with a bold name,
then the rest of the images in the directory
follow.  If there were new images, it
generates the alert messages, both emails and
text messages.
*/
<html>
<head>
<title>Ireland A'07 MQP Project - Wireless Surveillance System</title>
<meta http-equiv='refresh' content='30' url='index.php'>
</head>
<h1 align="center" style="color:#003333">Captured Images</h1>
<h4 align="center" style="color:#F0F0F0"<a href="email.html">
Add an Email Address</a> <a href="mobile.html">Add a Mobile</a></h4>
<hr>
<body>
<body bgcolor="#F0F0F0">
<p style="color:#003333">



<?php
/*
This code required for sending emails with attachments.
It is a part of PHP Extension and Application
Repository (PEAR).  One can find more information about
these files at http://devzone.zend.com/node/view/id/1052.
```

```
*/
include("Mail/mime.php");
include("Mail.php");

function emails_to_list($v1,$v2)
{
return $v1 . "," . $v2;
}

$new_images_found = 0; //flag for indicating receipt of new images
$local_files = scandir('.'); //array of all files in the local directory
$imgs = fopen('images.txt','r');
$image_names = file('images.txt'); //array of old images
fclose($imgs);
$max = count($local_files);
$all = count($image_names);
$new = array(''); //array to hold the names of new images

//compare files in the local directory to known images
for ($i=0; $i<$max; $i++){
$is_new = 1;
$temp = rtrim($local_files[$i]);
for ($m=0; $m<$all; $m++){
if (strcmp($temp, rtrim($image_names[$m]))){
//not the same, therefore new
}else{
//the same, therefore not new
$is_new = 0;
}}
if($is_new==1){
//it's not on the list, but it may not be an image
if (strpos($temp, '.jp')){ //is it a JPEG/JPG?
array_push($new, $temp); //add to array of new images
$new_images_found = 1; //set new flag
}}}

array_shift($new); //remove the first entry, which was just empty
$new_num = count($new);
//upload all new file first, with bold name
```

```
if ($new_num > 0){
asort($new);
for ($j = 0; $j<$new_num; $j++){
?>
<center/>
<img align="center" src="<? echo $new[$j]; ?>"/>
<?
$tmp = $new[$j];
echo '<br/>'; ?>
<b>
<?
echo $tmp .'<br/><br/><br/>'; ?>
</b>
<?
}}

asort($image_names); //sort, then upload newest images first
$img_num = count($image_names);
for ($k = 0; $k < $img_num; $k++){
?>
<center/>
<img align="center" src="<? echo $image_names[$k]; ?>"/>
<?
$tmp = $image_names[$k];
echo '<br/>'. $tmp .'<br/><br/><br/>';
}

//add new images to the list of known images
$to_write = fopen('images.txt','a');
for ($n = 0; $n<$new_num; $n++){
fwrite($to_write, $new[$n]."\n");
}
fclose($to_write);

if ($new_images_found){
//sending alert messages
//first text message, no attachments
$text_message = new Mail_mime();
     $mobile_addresses = fopen('mobiles.txt','r');
```

```
     $mobile_array = file('mobiles.txt');
     fclose($mobile_addresses);
     $mobiles = array_reduce($mobile_array,'emails_to_list');
$text = "New activity in the car park.  Check the website.";
     $body = $text_message->get();
$text_message->setTXTBody($text);
     $extraheaders = array("From"=>"carpark@ul.ie",
     Subject"=>"UL Security Alert!");
     $headers = $text_message->headers($extraheaders);
     $mail_mobile = Mail::factory("mail");
     $mail_mobile->send($mobiles, $headers, $body);

//second emails, attachment of all new images
     $email_message = new Mail_mime();
     $text = "Foundation Car Park alert system";
     $email_message->setTXTBody($text);
for ($l = 0; $l<$new_num; $l++){
$tmp = $new[$l];
$email_message->addAttachment($tmp);
}
     $body = $email_message->get();
     $extraheaders = array("From"=>"carpark@ul.ie",
     "Subject"=>"Alert!");
     $headers = $email_message->headers($extraheaders);
     $email_addresses = fopen('email_addresses.txt','r');
     $emails_array = file('email_addresses.txt');
     fclose($email_addresses);
     $emails = array_reduce($emails_array,'emails_to_list');
     $mail_ppl = Mail::factory("mail");
     $mail_ppl->send($emails, $headers, $body);
     }
?>
</p>
</body>
</html>
```

```
/*
Base Station Code - email.html
Ireland A07 MQP - Efficient Surveillance System
Advisors: Prof. Vaz, Prof. Brown
Students: Matt Conway, Phong Dam, Janelle Tavares*
*author of this code

Summary:
This file contains code for the front end of a webpage.
This webpage allows a user to input an email address
so that user would get notification and attached
jpeg files of new activity captured by the surveillance
system.
*/
<html>
<head>
<title>Ireland A'07 MQP Project - Wireless Surveillance System</title>
</head>
<h1 align="center" style="color:#003333">Add an Email Address</h1>
<h4 align="center" style="color:#F0F0F0"
<a href="http://www.ul.ie/ULMQP07">Home</a> </h4>
<hr>

<body>
<body bgcolor="#F0F0F0">
<p style="color:#003333">

<br/><br/><br/>
<center/>
<h2> Enter an email address. </h2>

<br/><br/>

<form name="input" action="new_email.php" method="post">
Email Address:
<input type="text" name="email">
<input type="submit" value="Submit">
</form>
```

```
</p>
</body>
</html>
```

```
/*
Base Station Code - new_email.php
Ireland A07 MQP - Efficient Surveillance System
Advisors: Prof. Vaz, Prof. Brown
Students: Matt Conway, Phong Dam, Janelle Tavares*
*author of this code

Summary:
This file contains code for the back end of a webpage.
When a user inputs the email address into the form on
the email.html page, the data becomes available to
this page.  It opens a file with email addresses,
adds the inputted address, and notifies the user
after making that change.
*/
<html>
<head>
<title>Ireland A'07 MQP Project - Wireless Surveillance System</title>
</head>
<h1 align="center" style="color:#003333">Add an Email Address</h1>
<h4 align="center" style="color:#F0F0F0"
<a href="http://www.ul.ie/ULMQP07">Home</a> </h4>
<hr>
<br/><br/><br/>
<center/>

<?php
$new_email = $_POST["email"];
$list = fopen('email_addresses.txt','a');
fwrite($list, $new_email."\n");
fclose($list);


echo $new_email . ' will be notified when the
security system detects new activity.';
?>
```

```
</p>
</body>
</html>
```

```
/*
Base Station Code - mobile.html
Ireland A07 MQP - Efficient Surveillance System
Advisors: Prof. Vaz, Prof. Brown
Students: Matt Conway, Phong Dam, Janelle Tavares*
*author of this code

Summary:
This file contains code for the front end of a webpage.
This webpage allows a user to input the email address
of a mobile device so that it would be notified of
new activity  captured by the surveillance system.
*/
<html>
<head>
<title>Ireland A'07 MQP Project - Wireless Surveillance System</title>
</head>
<h1 align="center" style="color:#003333">Add a Mobile</h1>
<h4 align="center" style="color:#F0F0F0"
<a href="http://www.ul.ie/ULMQP07">Home</a> </h4>
<hr>

<body>
<body bgcolor="#F0F0F0">
<p style="color:#003333">

<br/><br/><br/>
<center/>
<h2> Enter the email address of a mobile device. </h2>

<br/><br/>

<form name="input" action="new_mobile.php" method="post">
Email Address of Mobile:
<input type="text" name="mobile">
<input type="submit" value="Submit">
</form>
```

```
</p>
</body>
</html>
```

```
/*
Base Station Code - new_mobile.php
Ireland A07 MQP - Efficient Surveillance System
Advisors: Prof. Vaz, Prof. Brown
Students: Matt Conway, Phong Dam, Janelle Tavares*
*author of this code

Summary:
This file contains code for the back end of a webpage.
When a user inputs the email address of a mobile
device into the form on the mobile.html page, the data
becomes available to this page.  It opens a file
with addresses for mobile devices, adds the inputted
address, and notifies the user after making that change.
*/
<html>
<head>
<title>Ireland A'07 MQP Project - Wireless Surveillance System</title>
</head>
<h1 align="center" style="color:#003333">Add a Mobile</h1>
<h4 align="center" style="color:#F0F0F0"
<a href="http://www.ul.ie/ULMQP07">Home</a> </h4>
<hr>
<br/><br/><br/>
<center/>

<?php
$new_mobile = $_POST["mobile"];
$list = fopen('mobiles.txt','a');
fwrite($list, $new_mobile."\n");
fclose($list);

echo $new_mobile . ' will be notified when
the security system detects new activity.';
?>

</p>
</body>
</html>
```

```
#define _WIN32_WINNT 0x0400
#include <Windows.h>
#include <stdio.h>
#include <tchar.h>
#include <wchar.h>
#include <string.h>
#include <unistd.h>
#include <sys/param.h>
#include <sys/types.h>
#include <time.h>
#include "ftplib.h"

#define BUFSIZE MAX_PATH

using namespace std;

//this function returns 1 if the given string is in the array
//of size 'length', and returns 0 if it was not
int is_known(char* check_this, char** in_this, int length){
int i;
int check;
char** array_stepper = in_this;

for (i=0;i<length;i++){
check = strcmp(*array_stepper, check_this);
if(!check){
return 0;
}
array_stepper++;
}
return 1;
}
/*
Base Station Code - baseStation.cpp
Ireland A07 MQP - Efficient Surveillance System
Advisors: Prof. Vaz, Prof. Brown
Students: Matt Conway, Phong Dam, Janelle Tavares*
*author of this code
```

```
This code runs the part of the base station between the mesh
network team's portion and the web server.  This code will
continually check a specified folder for new .jpg files.
When it finds a new file, it sends it to our file share
on the web server with FTP.
Special note about the FTP library, ftplib.  We downloaded
our version from http://nbpfaus.net/~pfau/ftplib/.
The FTPlib.dll was broken and had to be rebuilt before we
were able to implement that library.
*/
int main(){
const int max_files = 10; //estimated number of new images in directory
const int max_known = 20; //estimate number of known images
const int max_filename_length = 21; //with null terminator
const int time_interval = 10;
//time to wait before checking for new images again
const char* baseDir = "base"; // base station image directory
const char* back = ".."; //string used to return to previous directory
const char* imageFile = "images.txt"; //file with names of images
int i; //counter
int num_known = 0; //current number of known images
int num_new = 0; //number of new images
char** holder; // holds address of allocated memory
int first = 1; //first time through program flag
char* checking;
//holder of current file name in the directory to check if it's new
int result; //holds comparison information for current file
char** push_known; // pointer to location of next spot for known image name
char** push_new; // pointer to location of next spot for new image name
char** next_name; //holder for pointers in image_names array
time_t last_time;
time_t this_time;
// used to activate loop again after 'time_interval' seconds
//intiate variables for searching base station directory for new images
WIN32_FIND_DATA FindFileData;
    HANDLE hFind = INVALID_HANDLE_VALUE;
    DWORD dwError;
    LPTSTR DirSpec;
    size_t length_of_arg;
```

```
    INT retval;
    LPCTSTR find = "*.jpg";

last_time = time(NULL); //initiate count for 'time_interval' seconds

//allocate memory for file names in base station directory
DirSpec = (LPTSTR) malloc (BUFSIZE);

//allocate memory for array of known image names
holder = (char**) calloc(max_known, sizeof(char*));
    char** image_names = holder;
    char* array_stepper = *holder;
    for(i = 0; i < max_known; i++){
    *holder = (char*)calloc(max_filename_length, sizeof(char));
    holder++;
    }

    //allocate memory for array of new image names
    holder = (char**) calloc(max_files, sizeof(char*));
    char** new_images = holder;
    array_stepper = *holder;
    for(i = 0; i < max_files; i++){
  *holder = (char*)calloc(max_filename_length, sizeof(char));
    holder++;
    }

    //move current working directory to the base station directory
int check = chdir(back);
check = chdir(back);
check = chdir(baseDir);

//initialize the FTP library
FtpInit();
netbuf* control;

//begin routine, which will be repeated every 'time_interval' seconds
this_time = time(NULL);
while(difftime(this_time, last_time) < time_interval){
```

```
 if( DirSpec == NULL )
 {
    printf( "Insufficient memory available\n" );
    retval = 1;
    goto Cleanup;
 }

 length_of_arg = strlen(find);
 strcpy(DirSpec, find);
// Find the first file in the directory.

if(first==1){
 //populate the array the first time around,
 //as a means for comparison later
printf("First time looking through the directory... \n");
next_name = image_names; //set stepper to the beginning of the array
 hFind = FindFirstFile(DirSpec, &FindFileData); // get first file name

 if (hFind == INVALID_HANDLE_VALUE)
 {
    retval = (-1);
 }else{
 strcpy(*next_name, FindFileData.cFileName); //add file to known
 next_name++; //increment array pointer
 num_known++; //increment number of known images
  while (FindNextFile(hFind, &FindFileData) != 0)
  // get remaining file names
  {
             strcpy(*next_name, FindFileData.cFileName);
 next_name++;
 num_known++;
  }
  dwError = GetLastError();
  FindClose(hFind);
  if (dwError != ERROR_NO_MORE_FILES)
  {
     _tprintf (TEXT("FindNextFile error. Error is %u.\n"),
              dwError);
 retval = (-1);
```

```
     goto Cleanup;
     }}
    retval  = 0;
    first = 0;
}else{
    hFind = FindFirstFile(DirSpec, &FindFileData);//get first file name

    if (hFind == INVALID_HANDLE_VALUE)
    {
       retval = (-1);
    }else{
    checking = FindFileData.cFileName; //save first file name
    result = is_known(checking,image_names,num_known);
    //is file in array of known images?
    if(result){//if new...
    num_known++;
    num_new++;
    push_known = image_names + (num_known - 1);
    //get location of next spot in array
    strcpy(*push_known, checking);
    //add to array of known images
    push_new = new_images + (num_new - 1);
    //get location of next spot in array
    strcpy(*push_new, checking);
    //add to array of new images
    printf("New file found, %s \n", checking);
    }
     while (FindNextFile(hFind, &FindFileData) != 0)
     //do it again for the rest of the files
     {
                checking = FindFileData.cFileName;
    result = is_known(checking,image_names,num_known);
    if(result){
    num_known++;
    num_new++;
    push_known = image_names + (num_known - 1);
    strcpy(*push_known, checking);
    push_new = new_images + (num_new - 1);
    strcpy(*push_new, checking);
```

```
    printf("New file found, %s \n", checking);
    }}
     dwError = GetLastError();
     FindClose(hFind);
     if (dwError != ERROR_NO_MORE_FILES)
     {
        _tprintf (TEXT("FindNextFile error. Error is %u.\n"),
                 dwError);
    retval = (-1);
     goto Cleanup;
     }}
     if(num_new){
     result = FtpConnect("www.ul.ie", &control);
    result = FtpLogin("ULMQP07", "serum_glossy.bye", control);
    push_new = new_images;
    for(i=0;i<num_new;i++){
    //send file and save as same name
     result = FtpPut(*push_new,*push_new,FTPLIB_IMAGE, control);
     printf("%s sent. \n", *push_new);
     push_new++;
    }
    FtpQuit(control);
    num_new = 0;//reset number, as all new have been sent
     }
}
do{ //wait for time_interval to be over
this_time = time(NULL);
}while(difftime(this_time, last_time) < time_interval);
printf("Checking again.\n");
last_time = time(NULL); //reset time values
this_time = time(NULL);
}

Cleanup:
   free(DirSpec);
   free(image_names);
   return retval;
}
```