Remote Control Car using Wi-Fi

by

Wan Nursyamsiah Wan Jusoh
4583

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons) in
Information & Communication Technology

December 2006

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

t
TK
5103.2
.W244
2006

1) Wireless communication systems
2) IT/IS -- Thesis

CERTIFICATION OF APPROVAL

Remote Control Car Using Wi-Fi

By

Wan Nursyamsiah Binti Wan Jusoh

A project dissertation submitted to the
Information & Communication Technology Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION & COMMUNICATION TECHNOLOGY)

Approved by,

____

(Mr. Abdullah Sani Abdul Rahman)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
December 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

WAN NURSYAMSIAH BINTI WAN JUSOH

ABSTRACT

Wireless Fidelity, Wi-Fi utilize one of the IEEE 802.11 wireless standards to achieve a wireless network. In this project, author has to use Wi-Fi in controlling a Wi-Fi car. The problem statement, objectives and scope of studies for this project will be further explained in the first part of this report, the INTRODUCTION section. As for the scope of study, author explores Wi-Fi technology itself and the information gained was documented in LITERATURE REVIEW/THEORY chapter. The proposed METHODOLOGY used in this project is discussed in the next section, including the tools and software utilized in developing the system. As this project is meant to develop author understanding on wireless network, the results of this project are included in the RESULT AND DISCUSSION section. The last section, CONCLUSION AND RECOMMENDATION will conclude author activities throughout these two semesters and describe suggestions to enhance and overcome flaws of this project. All references of this project can be referred in the REFERENCES list.

ACKNOWLEDGEMENT

I am indebted to many individuals who helping me during these two semesters in finishing this project where the presence are the essence to make this project successful. They are the people of my respects who involve directly and indirectly throughout the progress of this project.

First and foremost, my warmest gratitude goes to my supervisor, Mr. Abdullah Sani Abdul Rahman for his unwavering support and guidance while completing this project. His generosity and help have been an inspiration to me. My sincere appreciation goes to the FYP Committee for all their patience and understanding in guiding and assisting me in this project.

A bunch of thanks to all Information & Communication Technology (ICT)/Business Information System (BIS), and Electrical and Electronics (EE) lecturers for their valuable expertise, guide and support in completing this project. Special thanks to Mr. Musa Yusof, the expert technician in robotic field who gave so generously of his time, opinions and efforts in guiding and assisting me throughout the completion of this project.

My special appreciation goes to all my friends in UTP for their priceless support and encouragement to fire up my motivation during the hard times I had to go through at some points in the period. Not forgotten to those who had share their knowledge and experience with me. The contributions you have made throughout the development process have been invaluable to me.

Last but not least, my deepest gratitude goes to my family members for their valuable advices, extra funding for the expensive hardware, constant love, unwavering support, and their understanding of me. Without all of them I would not go further like where I standing right now.

Again, thank you for your encouragement and continuous support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND NOMENCLATURES

IEEE	Institute of Electrical and Electronics Engineers
BSS	Basic Service Set
IBSS	Independent Basic Service Set
ESS	Extended Service Set
AP	Access Point
Wi-Fi	Wireless Fidelity
RF	Radio Frequency
HF	High Frequency
VHF	Very High Frequency
P-to-P	Peer-to-Peer
WLAN	Wireless Local Area Network
MAC	Medium Access Control
DCF	Distributed Coordination Function
ICMP	Internet Control Message Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
MCU	Microcontroller Unit
WEP	Wireless Equivalent Privacy

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND OF STUDY

This Final Year Project is focused on Wireless Fidelity (Wi-Fi) Technology to be integrated in a simple remote control car. It is based on the IEEE 802.11 specifications which provide coverage of up to 500 feet indoors and 1500 feet outside. Wi-Fi supports 1, 2, 5.5 and 11mbps speeds to carry data as well as voice communication. With these characteristics, this technology is the most suitable to be employed and indirectly solve the related problem statement in this project. Using electromagnetic waves, wireless LAN transmit and receive data over the air, minimizing the use of human resources and energy in patrolling areas.

1.2 PROBLEM STATEMENT

1.2.1 Problem Identification

Patrolling an area is not an easy task as we might think. Basically a security guard has to travel regularly through an area or long distance buildings in order to make sure the location is absolutely free from any dangerous activities or unexpected incidents. As the matter of fact, walking alone in the middle of the night still gives zero guarantees that the premises are safe. Furthermore, it is undeniable that this tour of duty requires a lot of energy and stamina which is tiring. This situation will lead to the unethical manners while working, for instance sleeping because of exhausted. Closed Circuit Television (CCTV) really helps the guard to monitor those areas. Unfortunately, those cameras are static or can be limitedly adjusted. A moving device that acts as the third eye to the guard can overcome this problem. Wi-Fi Car offers solutions that address the shortcomings of the traditional way of patrolling an area.

1.2.2 Significance of Project

With the use of ordinary radio frequency (49 MHz), it may not cover all specific areas. The same thing goes if infrared and Bluetooth technology are given priority. How about 3G technology? It is also not suitable since 3G is using licensed spectrum under specific service provider. The use of Wi-Fi technology seems the perfect response in developing the car.

1.3 OBJECTIVE AND SCOPE OF STUDY

The objectives of this project are:

(1) To study and familiarize author with Wi-Fi technology, devices and services.

(2) To model a remote control car by using Wi-Fi technology.

(3) To expose author to a different range of expertise as author will have to deal with circuit.

1.3.1 Feasibility of the Project within the Scope and Time Frame

At first glance, this project seems easy to be implemented as author just needs to follow ordinary remote controlled car's design. Actually dealing with different technology and devices make this project tougher than expected. In order to complete this project, a lot of research and practice need to be done. The project will be divided into two parts and each part will have different subsections. Author has to prepare backup plans of each phase in case of unexpected result turns out.

The two main parts of this project are:

(a) Require a user to control a car by using his/her laptop. Author should be able to include the basic car movements (forward, reverse, left, and right) in her program. The instruction will be transmitted to the receiver in order to control motors of the car, and the car will move accordingly.

(b) Require author to integrate web cam with the car. The live video will be streamed to the user's laptop/PDA.

Part B will be ignored in this project. The feasibility of this project can be evaluated through technical and schedule factors.

Technical Feasibility

This project can be completed with technical guidance from IT and EE technicians. The author also joins an internet group which discusses the related issues in developing Wi-Fi firmware for microcontroller. It improves author understanding on this matter and indirectly affect author skills in programming.

Schedule Feasibility

The author has scheduled the time given according to the suggested milestone that can be referred from the Final Year project Guidelines. Please refer to Appendix A for the project timeline. It is feasible to complete the project within the given time frame if the resources are available.

CHAPTER 2
LITERATURE REVIEW AND THEORY

2.1 INTRODUCTION TO Wi-Fi

Wi-Fi radio is using one of the wireless communications that works with 802.11 standard. This unique standard was created by IEEE and it uses radio transmissions to connect computer devices to a network or to each other at distances of up to about 100 meters. The reliable coverage range for 802.11 WLANs depends on several factors, including data rate required and capacity, sources of RF interference, physical area and characteristics, power, connectivity, and antenna usage. Theoretical ranges are from 29 meters (for 11 Mbps) in a closed office area to 485 meters (for 1 Mbps) in an open area. However, through empirical analysis, the typical range for connectivity of 802.11 equipment is approximately 50 meters (about 163 ft.) indoors. A range of 400 meters, nearly ¼ mile, makes WLAN the ideal technology for many campus applications. It is important to recognize that special high-gain antennas can increase the range to several miles. According to Wikipedia.Com; Wifi is a brand originally licensed by the Wi-Fi Alliance to describe the underlying technology of wireless local area networks (WLAN) based on the IEEE 802.11 specifications. Wi-Fi was developed to be used for mobile computing devices such as laptops, in LANs but is now increasingly used for more applications including Internet and VoIP phone access, gaming, and basic connectivity of consumer electronics such as televisions and DVD players, or digital cameras.

Wi-Fi technology is using mesh topology, where collection of wireless devices maintaining RF connectivity to create a seamless (flawless) path for data packets to travel. Wi-Fitechnology.com in it website describes that a wireless mesh network resembles an idealized version of a top-level Internet backbone in which physical location is less important than capacity and network topology. In the wireless mesh environment, a network can be pictured as a collection of access points, routers, or end users (equipped with wireless receiver/transmitters) that are free to move randomly but maintain a reliable communication that sends and receive messages.

IEEE 802.11 defines two kinds of services that are Basic Service Set (BSS) and Extended Service Set (ESS). BSS is made of stationary or mobile wireless stations and a central base station, known as access point (AP). Another topology is know as ad hoc network, is meant to easily interconnect mobile devices that are in the same area (e.g., in the same room). In this architecture, client stations are grouped into a single geographic area and can be Internet-worked without access to the wired LAN (infrastructure network). The interconnected devices in the ad hoc mode are referred to as an independent basic service set (IBSS). It is a stand alone network and cannot send data to other BSSs. In this architecture, stations can perform a network without AP as long as they agree to participate in the network. In short, user devices communicate directly with each other in a peer-to-peer manner. ESS is wider than BSS as it is created when two or more BSSs with APs are connected through distributed networking, and this is called an infrastructure network.

The fundamental structure of a Wireless Local Area Network (WLAN) is the peer-to-peer or peer-to-multipoint communication between two wireless devices, called ad hoc. The purpose is forming a collection of wireless devices that maintain connectivity with each other while transferring or routing data in a random manner. Most wireless LAN uses spread spectrum technology to transmit and receive data. Spread spectrum was developed and used by military, because of its reliability, security and mission-critical communication system of operation.

Spread-spectrum is designed to trade bandwidth efficiency for reliability, integrity and security. More bandwidth is consumed than that of other radio transmission system. The trade-off produces a signal that is, in effect louder and easier to detect, providing the receiver knows the parameter of the spread-spectrum signal being broadcast. Any other receive not tuned to the right frequency will view transmission as background noise. There are two types of spread-spectrum technologies and they are frequency hopping and direct sequence.

In peer-to-peer (P-to-P) configuration, each wireless link replaces a single communication cable and can converse reliably as long as the two end points are close enough to escape the effects of Radio Frequency (RF) interference or signal loss. To ensure maximum RF performance, the Wi-Fi cells incorporate high-powered radios operating at the maximum allowed regulatory limits (1W) in conjunction with high-gain omni directional (7.4 dBi) or directional antennas [2]. Compared with Bluetooth, Wi-Fi is more suitable to control a car in this project as it provides wider range and higher data rate.

2.2 BRIEF HISTORY

Motorola developed one of the first commercial WLAN systems with its Altair product. However, early WLAN technologies had several problems that prohibited its pervasive use. These LANs were expensive, provided low data rates, were prone to radio interference, and were designed mostly to proprietary RF technologies. The IEEE initiated the 802.11 project in 1990 with a scope "to develop a Medium Access Control (MAC) and Physical Layer (PHY) specification for wireless connectivity for fixed, portable, and moving stations within an area." In 1997, IEEE first approved the 802.11 international interoperability standard. Then, in 1999, the IEEE ratified the 802.11a and the 802.11b wireless networking communication standards. The goal was to create a standards-based technology that could span multiple physical encoding types, frequencies, and applications. The 802.11a standard uses orthogonal frequency division multiplexing (OFDM) to reduce interference. This technology uses the 5 GHz frequency spectrum and can process data at up to 54 Mbps. This history was taken from [17].

2.3 HOW WI-FI WORKS

According to cease-wire.co.uk, Wireless LAN use electromagnetic airwaves to communicate information from one point to another without relying on any physical connection. Radio waves are often referred ad radio carriers because they simply perform the function of delivering energy to a remote receiver. By superimposing the transmitted data onto the radio carrier, data can be accurately extracted at the receiving end. This is generally referred as modulation of the carrier by the information being transmitted. Once data is modulated, the radio signal occupies more than a single frequency, since the frequency or bit rate of the modulating information adds to carrier. Multiple radio carriers can exist in the same space at the same time without interfering with each other if the radio waves are transmitted on different radio frequencies. To extract data, a radio receiver tunes in one radio frequency while rejecting all other frequencies.

As we all know, a typical Wi-Fi setup contains one or more Access Points (APs) and one or more clients. An AP broadcasts its Service Set Identifier (SSID) via packets that are called beacons, which are broadcast every 100 ms. The beacons are transmitted at 1 Mbit/s, and are of relatively short duration and therefore do not have a significant influence on performance. Since 1 Mbit/s is the lowest rate of Wi-Fi it assures that the client who receives the beacon can communicate at least 1 Mbit/s. Based on the settings (e.g. the SSID), the client may decide whether to connect to an AP. Also the firmware running on the client Wi-Fi card is of influence. Say two APs of the same SSID are in the range of the client, the firmware may decide based on signal strength to which of the two APs will connect. The Wi-Fi standard leaves connection criteria and roaming totally open to the client. This is the strength Wi-Fi but also means that one wireless adapter may perform substantially better than the other. Since Wi-Fi transmits in the air, it has the same properties as non-switched Ethernet network.

2.4 THE WI-FI STANDARD

The 802.11 family currently includes six over-the air modulation techniques that all use the same protocol, and the most popular techniques are those defined by the b, a and g notations, which are described as in Table 1.

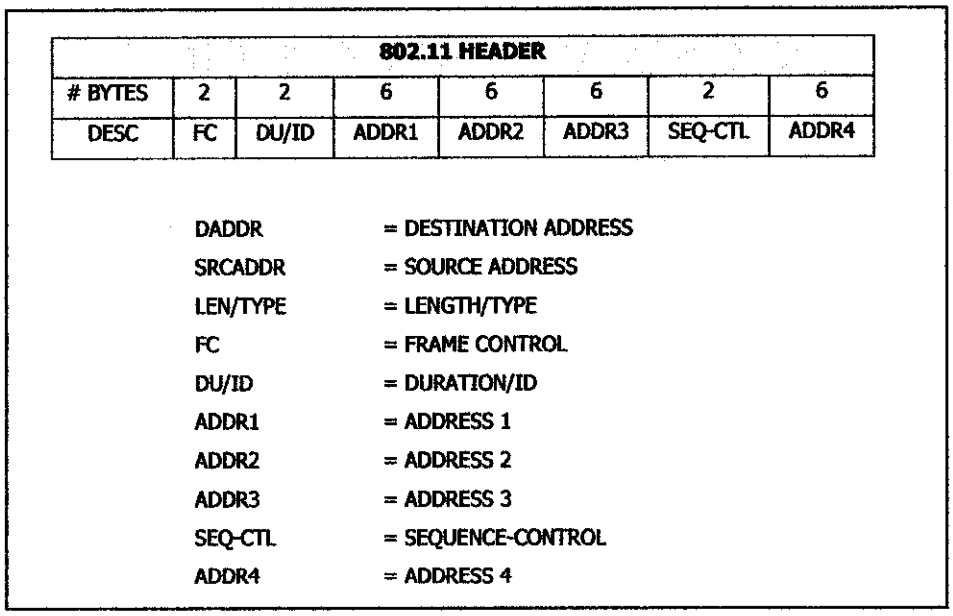
Technique	**Description**
802.11a	It operates at 5 GHz and can handle up to 54 megabits per second. Although faster than 802.11b, 802.11a is not as frequently used, especially in home and small office settings. 802.11a technology is not compatible with 802.11b technology.
802.11b	An addition to the 802.11 wireless LAN specification. 802.11b provides the potential for 11 Mbps (megabits per second) transmission (with a fallback to 5.5, 2 and 1 Mbps) at a radio frequency in the 2.4 GHz (gigahertz) band. 802.11b is a widely used wireless networking technology. 802.11b technology is not compatible with 802.11a technology.
802.11g	802.11g provides the potential for 54 Mbps (megabits per second) transmission (with several fallback transmission rates) at a radio frequency in the 2.4 GHz (gigahertz) band. 802.11g was developed as a higher-speed wireless technology (when communicating with other 802.11g devices) and is compatible with 802.11b devices. 802.11g technology is backward compatible with 802.11a technology.
802.11n	802.11n builds upon previous 802.11 standards by adding MIMO (multiple-input multiple-output). According to ritebrain.net, MIMO works by allowing two or more distinct signals to be transmitted over the same 802.11 radio channel at the same time with no interference. The additional transmitter and receiver antennas allow for increased data throughput through spatial multiplexing and increased range by exploiting the spatial diversity through coding

	schemes like Alamouti coding. It will feature speeds above 100 Mbps using multiple antennas and be backwards compatible with today's gear. It would be 4-5 times faster than 802.11g, and perhaps 50 times faster than 802.11b. The standardization progress is expected to be completed by the end of 2006.

Table 1: 802.11x capabilities

2.5 WI-FI PROTOCOL

Every transmission needs to follow protocol. The same thing goes to Wi-Fi technology. With that, author needs to know the 802.11 protocol in order to handle the frames. The general description of the 802.11 header is mentioned as below.



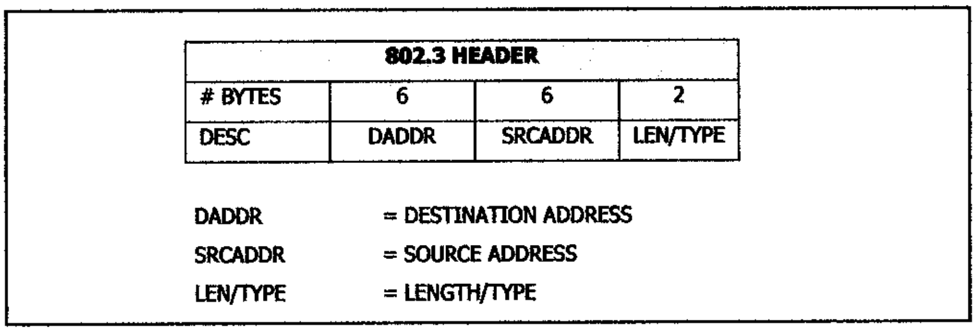
802.11 HEADER							
# BYTES	2	2	6	6	6	2	6
DESC	FC	DU/ID	ADDR1	ADDR2	ADDR3	SEQ-CTL	ADDR4

DADDR	= DESTINATION ADDRESS
SRCADDR	= SOURCE ADDRESS
LEN/TYPE	= LENGTH/TYPE
FC	= FRAME CONTROL
DU/ID	= DURATION/ID
ADDR1	= ADDRESS 1
ADDR2	= ADDRESS 2
ADDR3	= ADDRESS 3
SEQ-CTL	= SEQUENCE-CONTROL
ADDR4	= ADDRESS 4

Figure 1: 802.11 header

Address 1 is always the address of receiver (*destination address*), meanwhile Address 2 is the sender's address (*source address*). In a BSS, Address 3 can be either a source or destination address depending on which way the frame is flowing (to or away from AP). Address 4 is optional which only used in bridge mode. There is no direct 802.11 header support in the Wi-Fi code. The use of the 802.3 header helps this out. With the

802.3 header mode selected, the code will bypass the 802.11 header option in one of the functions. The 802.3 header can be referred to Figure 2.



802.3 HEADER			
# BYTES	6	6	2
DESC	DADDR	SRCADDR	LEN/TYPE

DADDR	= DESTINATION ADDRESS
SRCADDR	= SOURCE ADDRESS
LEN/TYPE	= LENGTH/TYPE

Figure 2: 802.3 header

2.5 COMPARISON WITH OTHER TECHNOLOGIES

The comparison of Wi-Fi and other technologies were described in Table 2. This comparison is very important in order to make sure the use of Wi-Fi suits this project very well.

Wi-Fi	3G
▪ To support wireless LAN. End user centric, decentralized approach to service provisioning	▪ 3g is a technology for mobile service provider
▪ Unlicensed spectrum	▪ Use licensed spectrum to provide wireless telephone coverage
▪ Possible to provide contiguous coverage over a wider area by using multiple base stations	▪ Include metropolitan area only but offers ubiquitous and continuous coverage.
▪ Broadband data service. Sufficient bandwidth to support real-time voice, data and streaming media.	▪ Broadband data service, support data rates from 384 kbps up to 2 Mbps (sufficient bandwidth to support real-time voice, data and streaming

▪ Higher bandwidth	media). ▪ Offers much narrower bandwidth but over a wider calling area. ▪ Advantage of facilitating QoS management
Bluetooth	**Infrared**
▪ Short range wireless communication, coverage up to 30 feet only ▪ Enabled devices operate in the 2.4 GHz radio frequency rage, but future versions will operate in 6-9 GHz range, eliminating the concern of interference from other wireless devices. ▪ Data transfer rates up to 3 Megabits (375 kilobytes) per second (slow transfer rate, suitable for transmitting smaller files such as text documents and cell phone contacts as well as lower quality images and audio). ▪ Built-in security	▪ No licenses needed ▪ Interference by sunlight, heat sources ▪ Low bandwidth

Table 2: Comparison of Different Technologies

2.6 ADVANTAGES OF Wi-Fi

Wireless LANs offer the following productivity, convenience and cost advantages:

Mobility: Provide LAN users with access to real time information anywhere at work and n home.

Installation Speed and Simplicity: Fast, easy and eliminate pull cables through walls and ceiling.

Installation Flexibility: Wireless technology allows the network to go where wire cannot go.

Reduced Cost-of-Ownership: Initial investment of Wi-Fi is higher than traditional LAN, but cheaper than 3G and overall installation expenses and life cycle costs can be significantly lower. Long term cost benefits area greatest dynamic environments requiring frequent moves and changes.

Scalability: Wi-Fi systems can be configured in a variety of topologies to meet the needs of specific applications and installations. Configurations are easily changed and ranged from peer-to-peer networks suitable for a small number of users to full infrastructure networks of thousands of users that enable roaming over a broad area.

CHAPTER 3
METHODOLOGY/PROJECT WORK

3.1 PROCEDURE IDENTIFICATION:

Methodology is steps taken in undergoing the research and project work. This research performs more on wireless characteristics (transfer rate, performance, etc) as well as developing the output of the problem initialized.

3.1.1 Process phase

In completing this project, author prefers to use prototyping-based and incremental methodology which author can perform analysis, design and implementation concurrently. These three phases are performed repeatedly in a cycle until the system is completed. This will give advantages to author as she can re-analyze, re-design, and re-implement the next prototype based on user comments.

Combining the first methodology with incremental prevents author from starting the whole project from zero point if any changes happen, hence just modified the current prototype. For each incremental phase, author will perform the subtasks identified in completing the whole project.

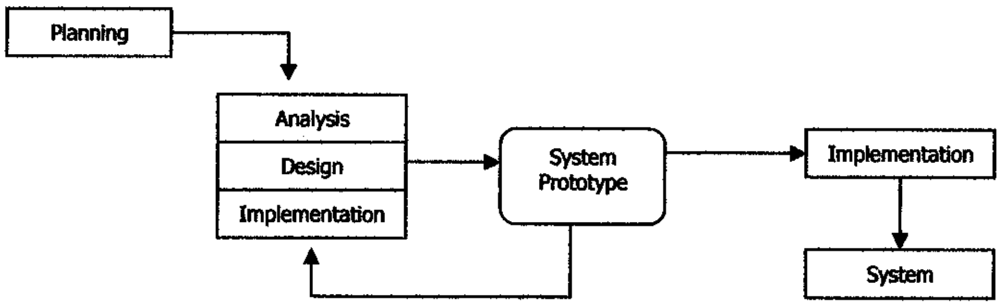


Figure 3: Methodology Process

Planning phase

In the planning phase, author chooses among several factors to set the application goals. The goals include anticipating and deciding on the target audience, purpose and objectives for the information. Besides, author had planned the activities in all stages and also the backup plans in case of unexpected results turn out.

Analysis Phase

In order to improve the application's quality, author did gathering and comparing about the application and functionality. This helps author in making decision during planning, design and implementing.

Design Phase

For the design phase, author concentrates on the limitation and constraint in developing this project. Besides, author did some research in designing the interface and emphasize on its clarity and user satisfaction.

Implementation Phase

In this phase, author builds the system using MP LAB IDE and VB6. Author also did testing and the testing is divided into three sections which are:

Unit Testing	: Testing of each small portion of the function
Module Testing:	Testing a few units together
System Testing:	Test the system as a whole.

3.2 PROJECT WORK

By referring to the problem statement, author divided the whole project into three main tasks. All these three tasks are not concurrently developed. Please refer to Appendix A for the project timeline. The tasks are briefly explained as below:

1. Remote Control System

   As the name implies, this system will be a remote control to control the Wi-Fi car. It will have the basic functions to control the car's movements.

2. Wi-Fi Car Firmware

   This firmware will be the brain to the Wi-Fi Car, which will be programmed into a microcontroller. It will read the received data sent by the transmitter (using Remote Control System) and ask the hardware to respond appropriately based on the command given.

3. Wi-Fi Car Body

   In this part, author needs to deal with circuit in developing the WiFi Car's body, which includes the H-bridge circuit and power regulator to control the direct current.

3.2.1 Remote Control System

After considering the working environment and the problem statement, author categorized the requirements into functional requirements and non functional requirements as below:

Functional requirements

1. Provide connection controls.
   a. To connect to the network
   b. To disconnect from the network

2. Type of movements
   a. Automatic
      - The car will keep moving until the stop command is triggered.
   b. Manual Control
      - The car will move in short distance only and it stops until it receives the new instructions.
3. The system should offer the movement controls. In this system, user has two ways to control the car. Instead of clicking the buttons, user should be able to use the system by pressing the appropriate keys assigned. The movement functions listed in the system are:
   a. Stop
   b. Forward
   c. Reverse
   d. Right
   e. Left

Non-Functional requirements

1. Security

Make sure a proper network connection between both ends, and include WEP is possible.

2. Performance

To prompt appropriate warning (e.g. lost control) if the signal strength slower and slower.

3. Usability

Make sure to provide a very good interface, thus eliminate confusion to the user. This software is expected to use 4 familiar keys, which are right, left, up and down arrows.

4. Integrity

Support easiness to copy the software to the other ends, or users. This software should able to work together with the other end.

5. Reliability

Maintain software reliability during the control. Let say a user presses the up arrow (forward) key, the system must send an appropriate signal to the other end (Wi-Fi car) and will move the car forward.

State Diagram

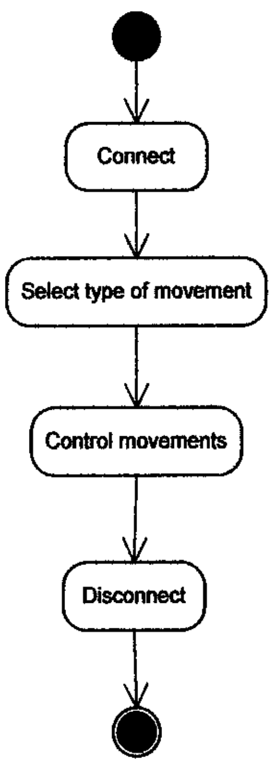


Figure 4: State Diagram

By referring to the above figure, user has to trigger the Connect function in order to control movement buttons. Furthermore, user can stop the connection anytime during the control.

Coding and Debugging

This system was developed using C++ language and utilized Microsoft Visual Studio. NET as the platform. The system then was redeveloped using VB 6.0 but still maintain the same functions. In this system, user has two ways to control the car. The first one is by pressing the button provided and the second one is by pressing appropriate keys assigned.

3.2.2 Wi-Fi Car Firmware

In this task, author needs to program microcontroller using MPLAB IDE and HI-TECH PICC-18 compiler. As author should focus on the program, author bought a wireless device with microcontroller, Airdrop-P. The default driver in Airdrop-P is using BSS mode and manages to handle Internet Control Message Protocol (ICMP), Transmission Control Protocol/Internet Protocol (TCP/IP) and User Datagram Protocol (UDP). Author hard coded the IP address of the car to be 192.168.0.151 and it will join the WIFICAR SSID. This firmware can be considered as the brain of the Wi-Fi car.

UDP is the main protocol used by author to interact between two ends of Wi-Fi hardwares. Below are the rough ideas of how this Airdrop-P works.

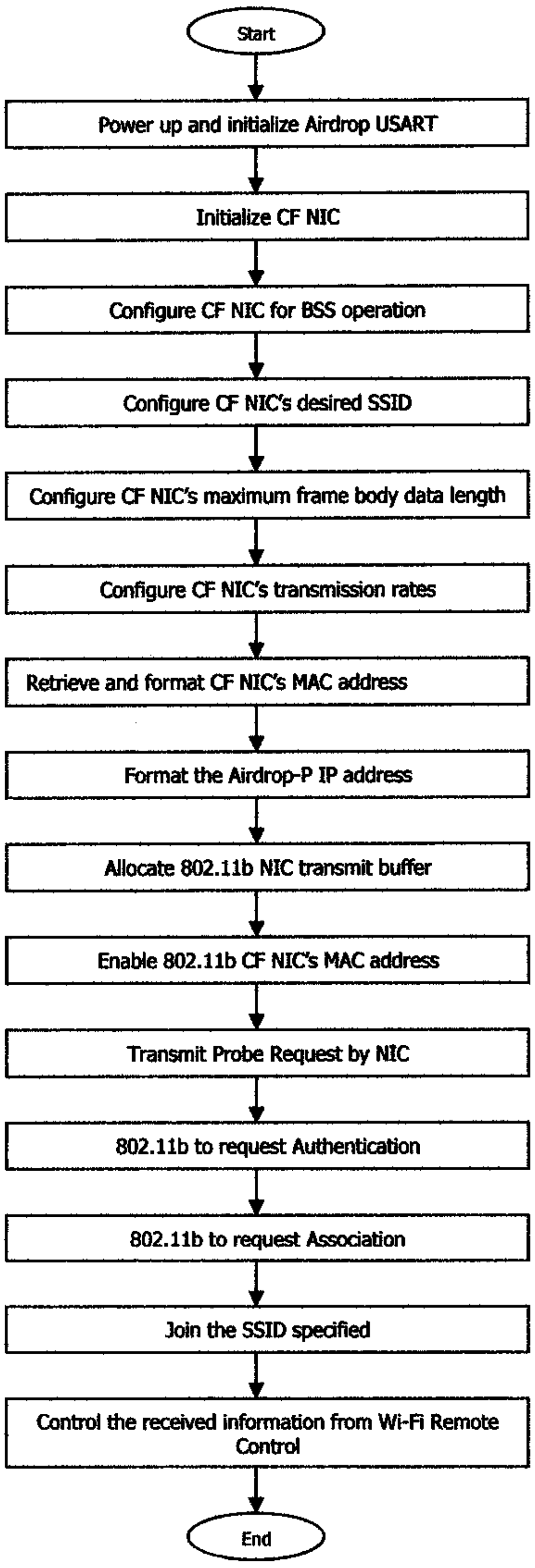

Start
Power up and initialize Airdrop USART
Initialize CF NIC
Configure CF NIC for BSS operation
Configure CF NIC's desired SSID
Configure CF NIC's maximum frame body data length
Configure CF NIC's transmission rates
Retrieve and format CF NIC's MAC address
Format the Airdrop-P IP address
Allocate 802.11b NIC transmit buffer
Enable 802.11b CF NIC's MAC address
Transmit Probe Request by NIC
802.11b to request Authentication
802.11b to request Association
Join the SSID specified
Control the received information from Wi-Fi Remote Control
End

Figure 5: Flow of Wi-Fi Car Firmware

As author only focused on UDP, author will elaborate on how the UDP works in Airdrop-P. As we all know this protocol will send and receive data but it did not care if it really gets where it is going or not. UDP modifies an application-generated message by tagging on a checksum, a source port number and a destination port number of its own before passing the UDP segment to IP for encapsulation. IP does its best to deliver the UDP segment since there is nothing within IP or UDP segment it is carrying to guarantee that the UDP segment will arrive intact.

Based on fact, UDP transmits a UDP datagram through a source port to a UDP recipient's destination port. The destination port number and destination IP address are used to route the UDP segment to the correct application once the segment arrives at its destination. Actually by using port numbers, various systems can use the same port simultaneously, which is called multiplexing. The word socket is used in the combination of the IP address and the port number. The interesting part of UDP is it does not require the establishment session with the remote host. No handshaking. No predetermined contact between UDP hosts. That is why UDP is called connectionless protocol. This will result in faster and efficient way to send a message.

UDP checksum is optional but this driver performs a checksum on every UDP datagram. The calculation of this checksum is a bit different compared to ICMP and IP checksum procedures. The UDP checksum includes some choice bytes from the IP header as well. The UDP checksum is calculated using below information:

- IP source address word
- IP destination address word
- IP protocol byte
- UDP length word
- UDP header
- UDP data

3.2.3 Wi-Fi Car Body

Apart of developing the software, author also needs to build the car body as the prototype to the Wi-Fi Car. The essential circuits/hardware needed in creating this car are:

1. Airdrop-P

   This circuit is equipped with microcontroller and Wi-Fi technology, which will be the core part of this project to receive signal sent from another Wi-Fi devices. Each and every command received will be translated to perform necessary action. The general specifications of Airdrop-P are:

   Full speed 11Mbps 802.11B Operation
   On-board +3.3V power supply
   Uses any PRISM2-3 CF Radio
   RS-232 Port
   PIC18LF8621 Microcontroller
   On-board ICSP for Program/Debug

2. H-bridge

   H-bridge has four switching elements at the "corners" of the H and the motor forms the cross bar. The key fact to note is that there are, in theory, four switching elements within the bridge. These four elements are often called, high side left, high side right, low side right, and low side left (when traversing in clockwise order). This circuit is useful in controlling the motor directions.

3. Power regulator

   This simple circuit is needed in order to maintain a constant voltage for the whole circuits. All voltage regulators operate by comparing the actual output voltage to some internal fixed reference voltage.

3.3 RESEARCH TOOLS

3.3.1 Hardware:

1. EDTP Airdrop-P

   This device is equipped with microcontroller and Wi-Fi technology, which will be the core part of this project to receive signal sent from another Wi-Fi devices. Each and every command received will be translated to perform necessary action (control motor direction, control camera, etc), thus can be considered as the brain of a simple car.

2. Motor

   Hardware to be connected to the microcontroller, and definitely will help to move the board based on the command sent from user.

3. Web cam

   This device will work as an eye to the user during controlling the car, which actually stream real live video to the user.

4. MPLAB ICD 2

   This device is intended to load firmware into the chip. As the name implies, it is actually the in- circuit debugger.

3.3.2 Software

1. Microsoft Visual Basic 6.0

   Software used to produce a simple remote control system.

2. Microsoft Project

   This software helps author in completing and maintaining the project timeline as the project activities are not running as planned and keep on changing due to unexpected problems arise. By using this software author can easily keep track her activities, and indirectly helps her to meet the task timeline on time.

3. Ethereal

   Help to capture packets to be analyzed before adding necessary functions in the Airdrop-P driver.

4. MPLAB IDE

   This program is compulsory to build program for microchip microcontrollers. A C compiler is used together with this software.

CHAPTER 4
RESULT AND DISCUSSION

4.1 RESULTS

As mentioned in the previous chapter, there are three major parts in this project. The first part is dealing with the Remote Control System, the second one is the Wi-Fi Car Firmware and last but not least is the Wi-Fi Car itself. In order to complete those three parts, strong knowledge on 802.11 frames and UDP protocol are required. The expected result of this project is shown in Figure 6.

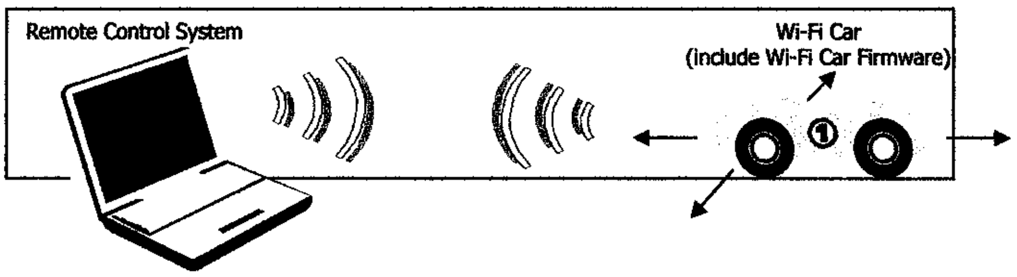


Figure 6: Expected Result

During the development, unexpected incident happen where the microcontroller resides in the Airdrop-P failed to be programmed. The developer of the Airdrop-P believes that the chip was damaged and the certain program memory failed to be read. Due to this problem, it seriously affects the development of Wi-Fi Car Firmware. This requires author to implement one of the contingency plans. List of contingency plans are as follows:

1. Rebuild the circuit
2. Connect the car by using laptop

At first author decided to rebuild the Airdrop-P circuit, but due to time constraint and hardware limitation, author had to continue with the second contingency plan. The circuit of the Airdrop-P can be referred to Appendix B.

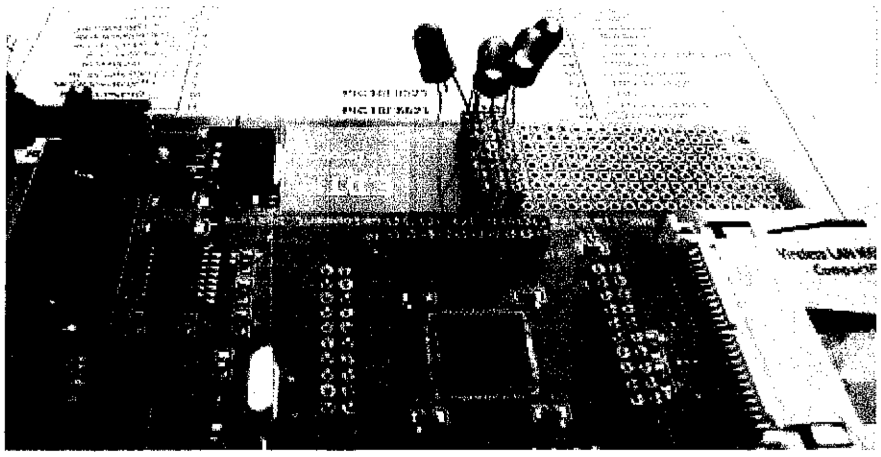
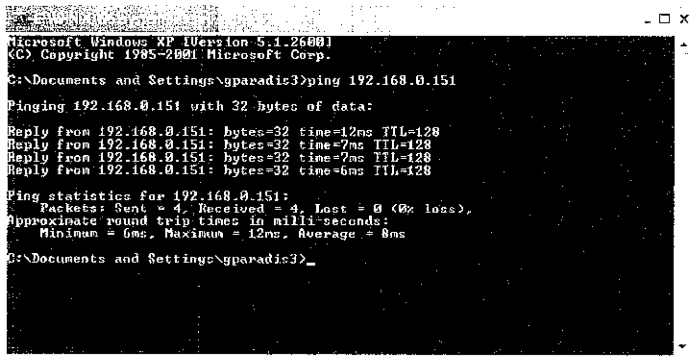


Figure 7: Power -up LEDs using Wi-Fi

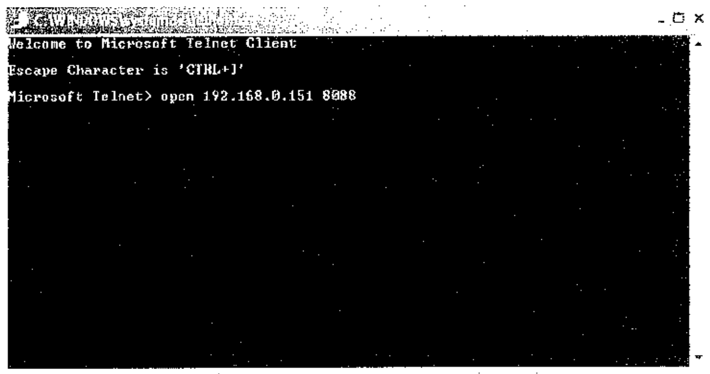
Before the microcontroller damaged, author successfully lighted up some LEDs from a laptop by using Remote Control System. Refer to Figure 7 for the output. Besides, author also tested the default function in the microcontroller, for instance ping the hardcoded IP address.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\gparadis3>ping 192.168.0.151

Pinging 192.168.0.151 with 32 bytes of data:

Reply from 192.168.0.151: bytes=32 time=12ms TTL=128
Reply from 192.168.0.151: bytes=32 time=7ms TTL=128
Reply from 192.168.0.151: bytes=32 time=7ms TTL=128
Reply from 192.168.0.151: bytes=32 time=6ms TTL=128

Ping statistics for 192.168.0.151:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 12ms, Average = 8ms

C:\Documents and Settings\gparadis3>_
```

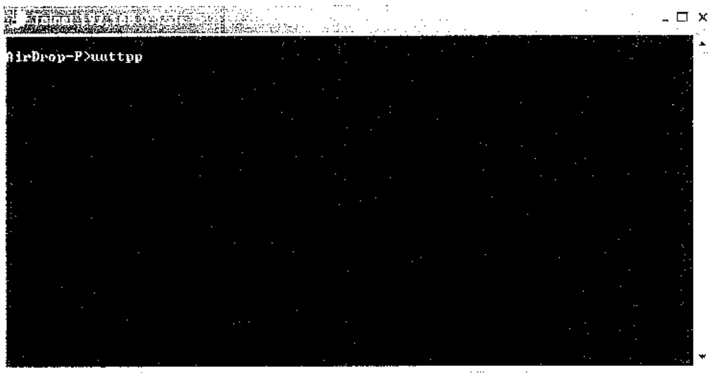
Figure 8: Pinging the Airdrop-P

The default driver also designed with minimal TCP/IP model. Below are the interfaces during the telnet session with Airdrop-P.



```
Welcome to Microsoft Telnet Client
Escape Character is 'CTRL+]'
Microsoft Telnet> open 192.168.0.151 8088
```

Figure 9: Telnet session with Airdrop-P



```
AirDrop-P>uuttpp
```

Figure 10: The echoed message of utp

By applying the contingency plan, there are no much changes that need to be done to the Remote Control System and the Wi-Fi Car body. The major adjustment is to eliminate the Wi-Fi Car Firmware and construct a new system (Wi-Fi Car System) and install it in a new laptop (any mobile devices with Wi-Fi will do). The expected result of using contingency plan can be referred to Figure 11.

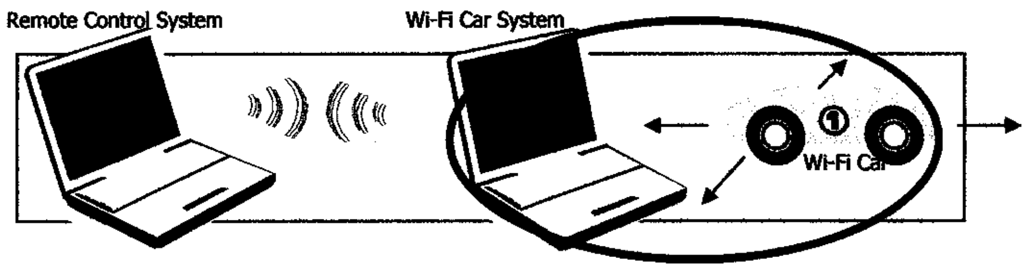


Figure 11: Expected Result using Contingency Plan

Therefore, the Wi-Fi Car must be connected to the laptop which has the Wi-Fi System in order to see the movements. Even though the new system (WiFi Car System) is not integrated in the Wi-Fi Car, the overall system is still implementing Wi-Fi technology. The screen shots of the final product can be referred to Figure 12, 13, 14 and 15. In this contingency plan, author stream a real live video from Wi-Fi Car System to the Remote Control System.

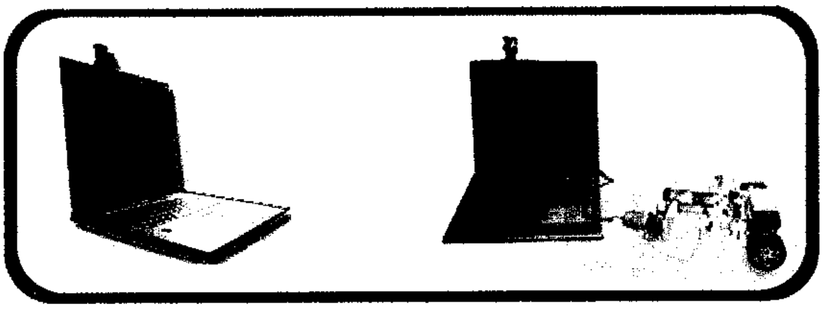
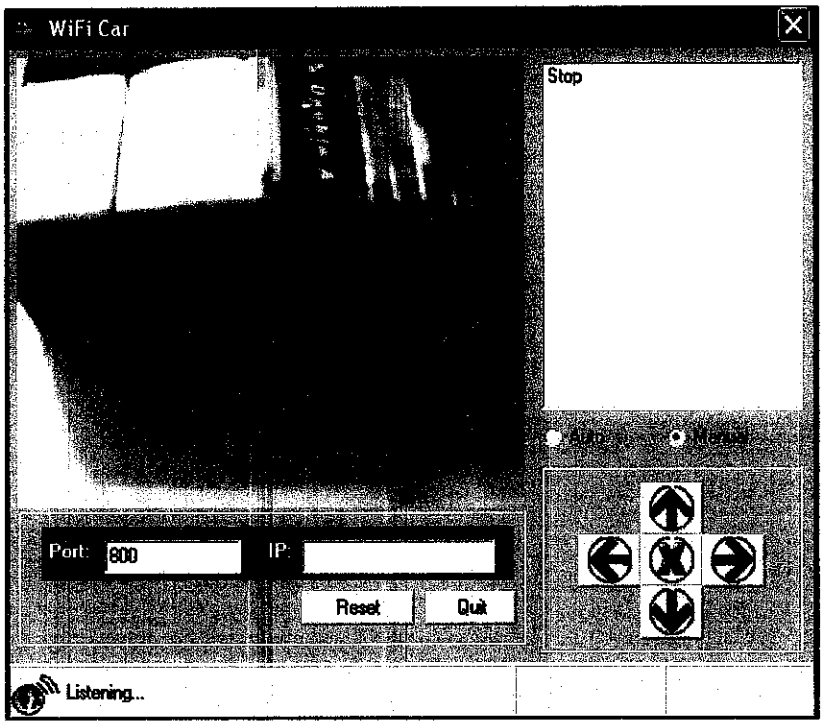


Figure 12: Final product



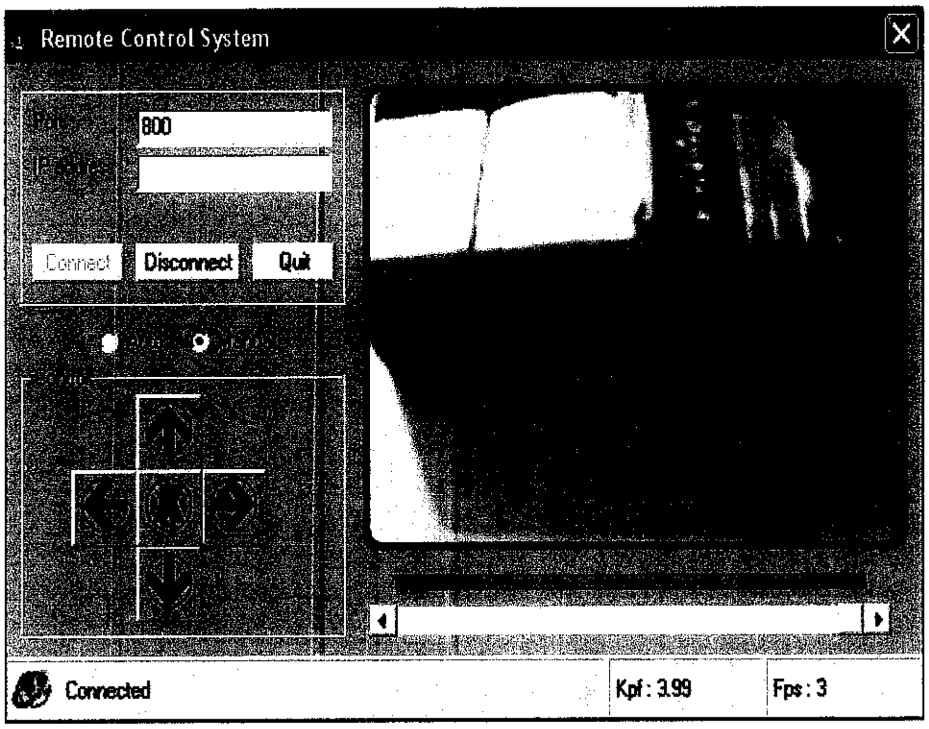
Stop

Port: 800

IP:

Listening...

Figure 13: The Interface of Wi-Fi Car System


Remote Control System

Port: 800

IP Address:

Connect | Disconnect | Quit

Connected | Kpf : 3.99 | Fps : 3

Figure 14: The Interface of Remote Control System

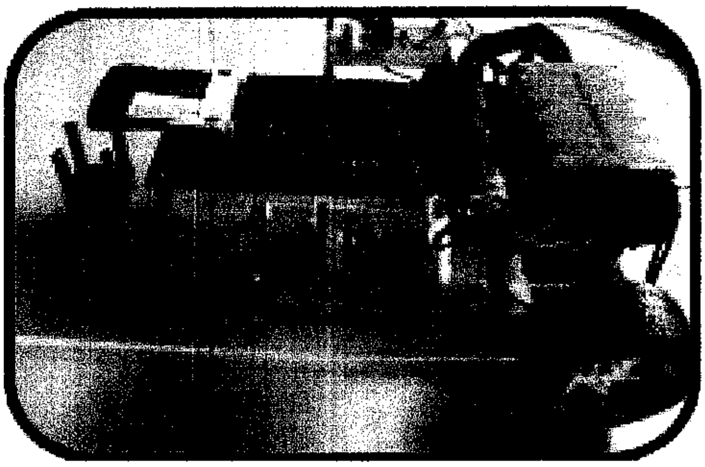


Figure 15: Wi-Fi Car

By referring to Figure 14, the arrow buttons in Remote Control System will trigger the appropriate command to the Wi-Fi Car System. These functions also can be controlled via key press. Refer to Table 3 for the button functions and the appropriate car movements.

Button	Keyboard	Function
↑	↑	Forward
↓	↓	Reverse
→	→	Right
←	←	Left
⊗	S	Stop

Table 3: Functions in Remote Control System

Wi-Fi Car System also has same functions as Remote Control Car System, but instead of sending command to other system, it directly sent the commands through parallel port to the car. The car will move accordingly based on the command received.

4.2 DISCUSSION

There are several problems arise, which affect the progress of this project. The first and foremost problem is related to the programmer to load a firmware into Airdrop-P. The use of MPLAB ICD 2 will ease author works; unfortunately UTP did not have this programmer. After some time, the EE department ordered this programmer but it haven't reach UTP yet and author has to wait until now. As an alternative way, author has to build her own programmer (refer to Figure 16) while waiting for ICD 2 by referring to [12] and [13]. Author did not sure how far this programmer fits Airdrop-P but in the middle of the development, author identified that the MCU type used in Airdrop-P is not listed in the software provided. Hence that forced author to wait for ICD 2. Since author sees no future in waiting the device for almost three months, the final decision was to buy the programmer online.

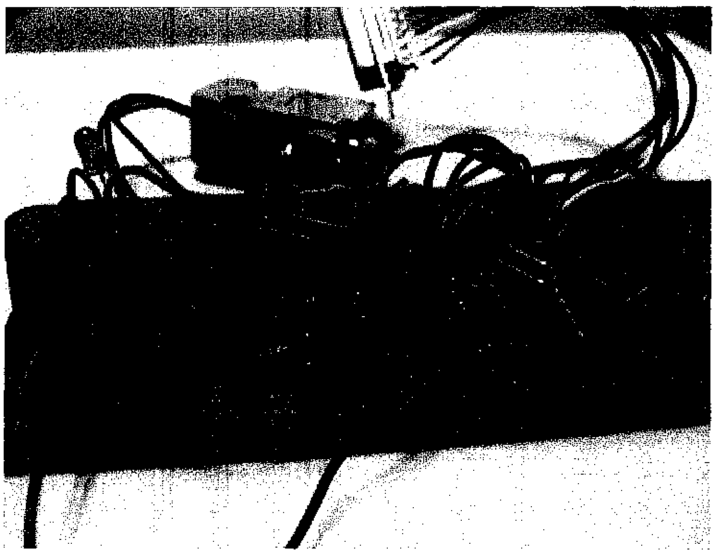


Figure 16: Programmer developed

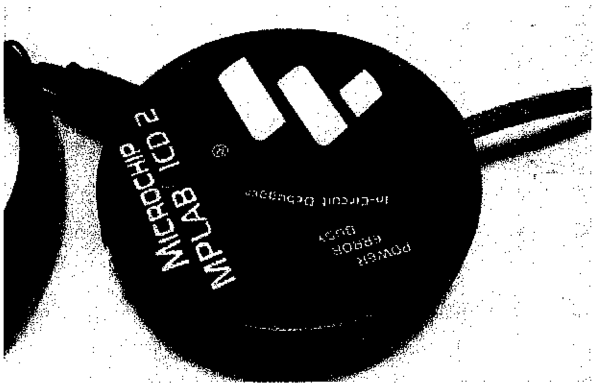


Figure 17: Microchip MPLAB ICD 2

The second problem is related to the default driver in Airdrop-P. As mentioned above, the driver is coded in BSS mode. The actual intention is to move wireless car in IBSS. It is undeniable that it can be done as one of the airdrop users had successfully run his own code in IBSS. The code is shared in the [10], but author really do not have much time to study on that as all works are delayed because of the programmer. With that, author prefers to stick to the default driver that is in BSS mode.

The third problem is related to the compiler. There are many compilers ready to be used and three of them are HI-TECH PICC-18, CCS C and C-18. Author prefers to use HI-TECH PICC-18, the same compiler as EDTP is using, but this one is only a demo version. The codes used in the default driver were mark as obsolete, which lead to fail in compiling the driver. As a temporary solution, author uses C-18, provided by microchip. This one is also a demo version and one of the Airdrop members was kind enough to share the modified code to be used in C-18. Since author already study the HI-TECH PICC-18 compiler, author decided to stick with it. After walkthrough the codes provided, author managed to solve the problems occurred and successfully produced the HEX file needed.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

As conclusion, author meets the objectives of this project. Even though author failed to complete the actual plan because of unexpected problem, author still utilized Wi-Fi technology in the contingency plan. Lots of knowledge in technical and non-technical has been gained. This project is important in solving the problem mentioned and also to broaden up Wi-Fi scope.

This project is one of the best platforms to achieve all the objectives and solve the issue. It is undeniable that author gain extra knowledge on Wi-Fi technology and indirectly help to develop author skills in programming field, especially in embedded programming and socket programming module. Besides, author also applied knowledge learned throughout eight semesters in UTP.

Furthermore, author realizes that she needs to tighten and patched her management skills due to the failure to meet some of the tasks timeline. It is understood that author has many other commitments such as tests, assignments, course projects and finals, but that are not the suitable reasons why she cannot complete some tasks in this project. Author strongly agrees that with a proper planning and wise time management, she can complete the all tasks on time.

5.2 RECOMMENDATION

After revising the whole project, author found that it requires a lot of efforts to fulfill all the requirements. Due to time constraints and some limitations, author dropped out some requirements. Author successfully controls the Wi-Fi Car by sending command from Remote Control System to Wi-Fi Car System. Below are the recommendations to this project:

1. Rebuild the Airdrop-P circuit
   Author recommends the continuer to rebuild the Airdrop-P circuit because it is costly to buy a new Airdrop-P device. The original circuit of Airdrop-P is using a surface mounted microcontroller, which is hard to be replaced if any problem occurred. Therefore, type of microcontroller also needs to be replaced. The schematic of Airdrop-P can be found in Appendix B.

2. Wired Equivalent Privacy(WEP) enabled
   Incorporate WEP features in the embedded software. WEP is a security protocol that is designed to provide a wireless local area network (WLAN) with a level of security and privacy by encrypting the data that is transmitted. It protects the vulnerable wireless connection between users (clients) and access points (APs).

REFERENCES

[1] Behrouz A. Forouzan, 2004, *Data Communications and Networking*, Third Edition, McGraw-Hill

[2] Malik Audeh, December 2004, *Metropolitan-Scale Wi-Fi Mesh Networks*, Institute of Electrical and Electronics Engineers (IEEE)

[3] Josh Broch, David A. Maltz, David B.Johnson, Yih-Chun Hu, Jojeta Jetcheva, *A performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols*, Institute of Electrical and Electronics Engineers (IEEE)

[4] Jinyang Li, Charles Blake, Douglas S.J. De Coute, Hu Imm Lee, Robert Morris, *Capacity of Ad Hoc Wireless Networks*, M.I.T Laboratory for Computer Science, IEEE

[5] Wi-Fi [Online], http://en.wikipedia.org

[6] Making the Wireless Home Network Connection in Windows XP Without a Router [Online], http://www.microsoft.com/windowsxp/using/networking/expert/bowman_02april08.mspx

[7] Virtual Reality in Medicine [Online], http://www.conniq.com/Windows-networking/Wi-Fi_ad-hoc_xp-setup_04.htm

[8] Fred Eady, 2005, Implementing 802.11 with Microcontrollers: Wireless Networking for Embedded System Designers, Newnes

[9] Ted Van Sickle, 2001, Programming Microcontrollers in C, Second Edition, LLH Technology Publishing

[10] Airdrop group [Online], http://groups.yahoo.com/group/airdrop_user/

[11] EDTP website [Online], http://www.edtp.com

[12] WISP628 [Online], http://www.pmb.co.nz/hobbycorner/pages/pro_vis0.htm

[13] WISP628 [Online], http://www.voti.nl/wisp628/index_1.html

[14] Microchip [Online], http://www.microchip.com

[15] RSMalaysia [Online], http://www.RSMalaysia.com

[16] Hi-TECH software [Online], http://www.htsoft.com

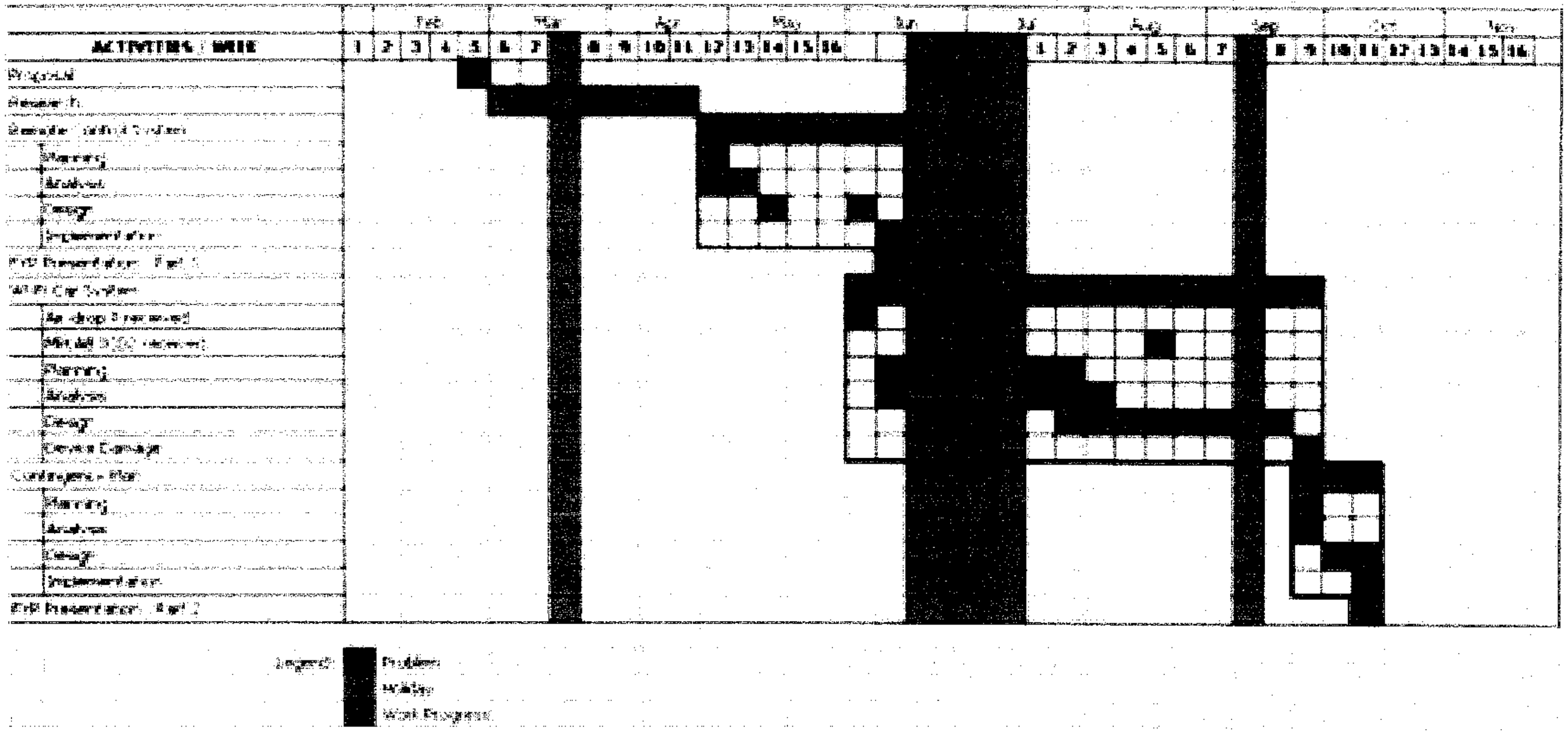
[17] people@morrisville[Online], people.morrisville.edu

REMOTE CONTROL CAR USING WI-FI

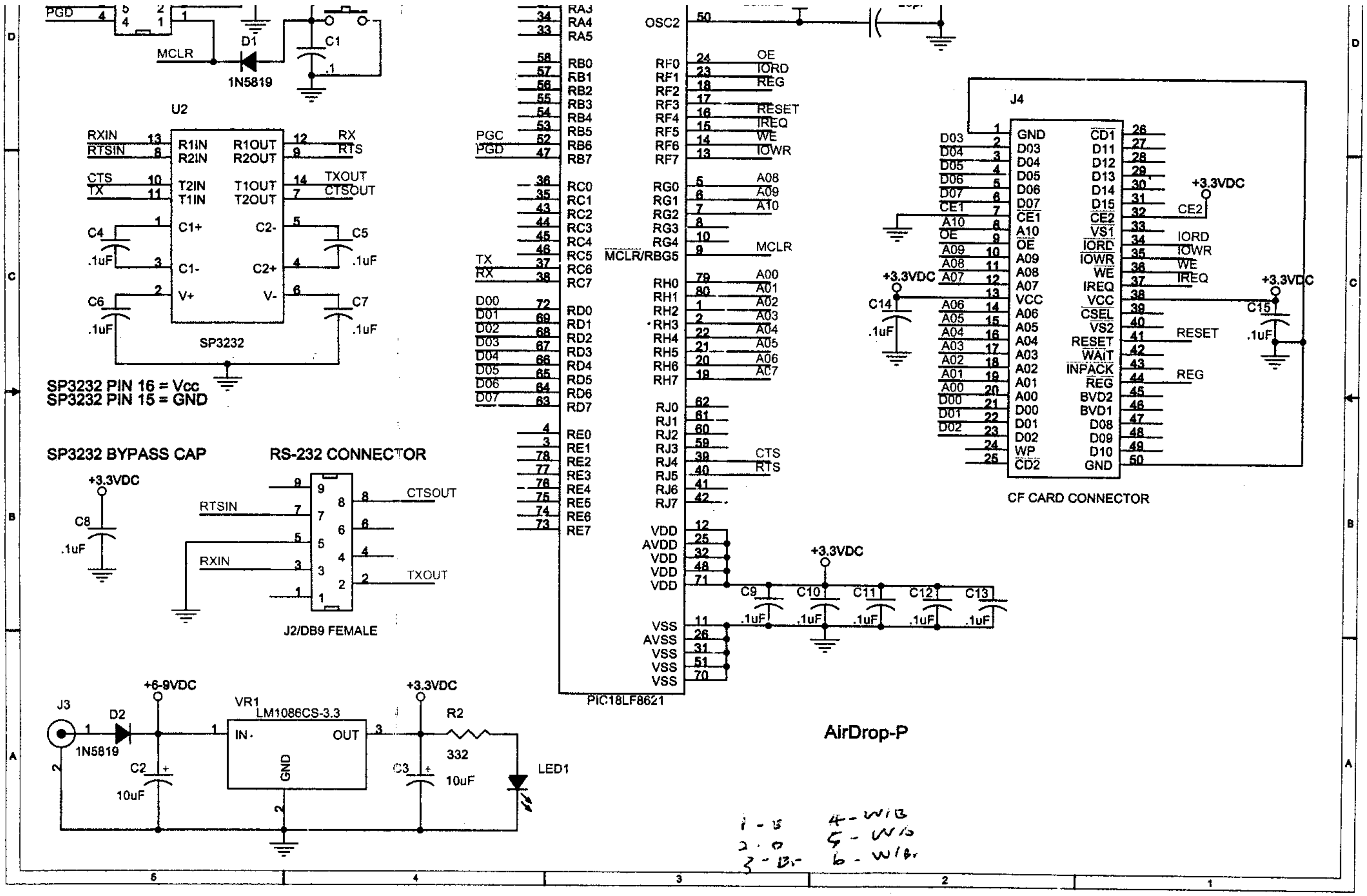
WAN NURSYAMSIAH BINTI WAN JUSOH

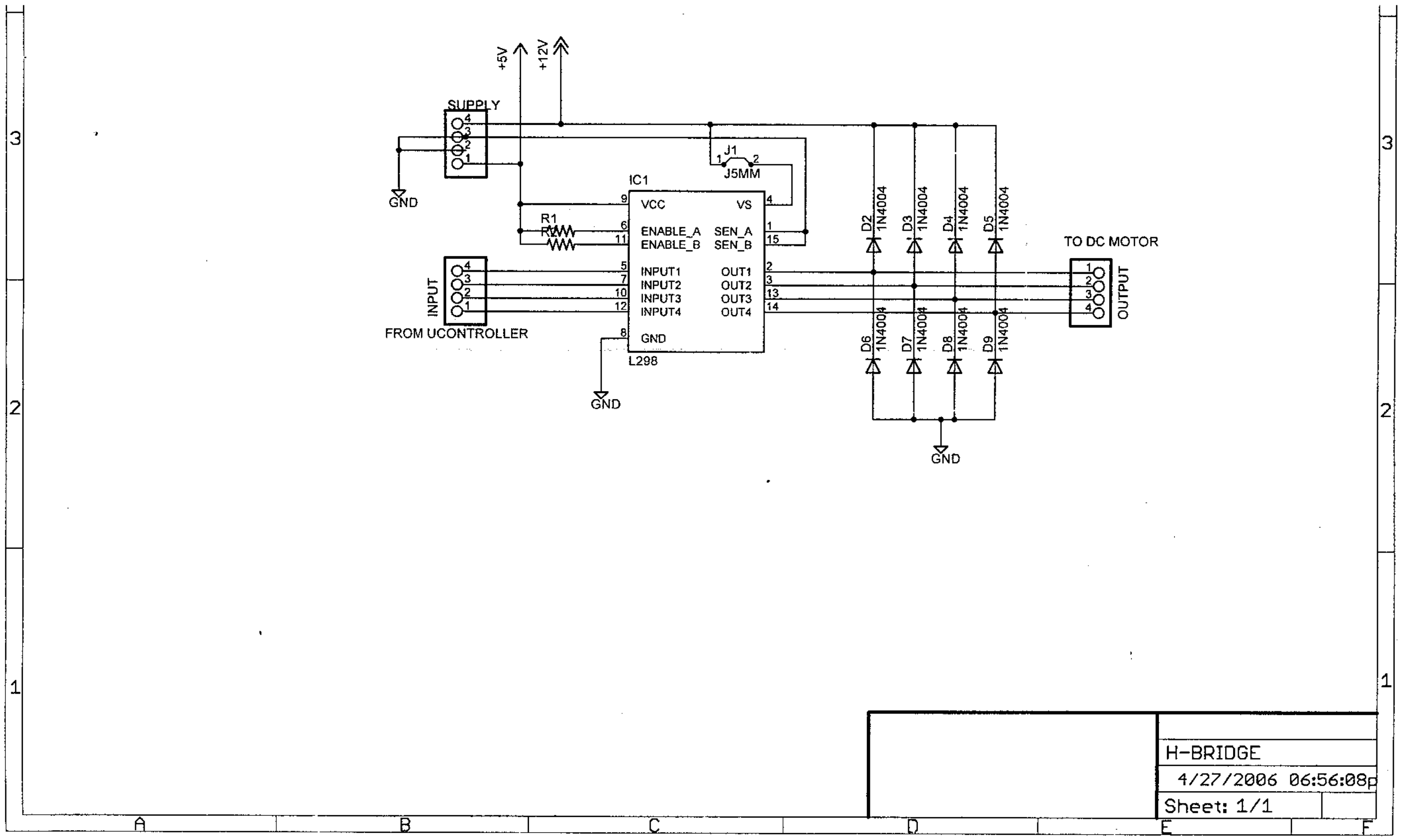
INFORMATION & COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNOLOGI PETRONAS
DECEMBER 2006

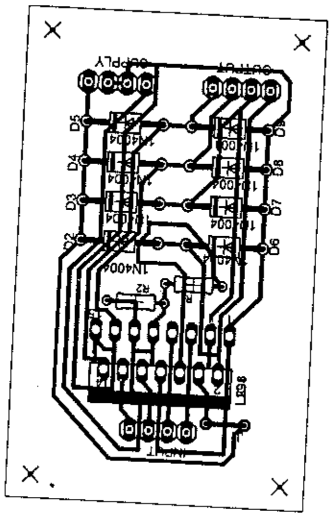
APPENDIX A


ACTIVITIES / WEEK
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Proposal
Research
Remote Control System
Planning
Analysis
Design
Implementation
FYP Presentation - Part 1
WiFi Car System
Planning
Analysis
Design
Contingency Plan
Planning
Analysis
Design
Implementation
FYP Presentation - Part 2
Legend:
Problem
Holiday
Work Progress

APPENDIX B

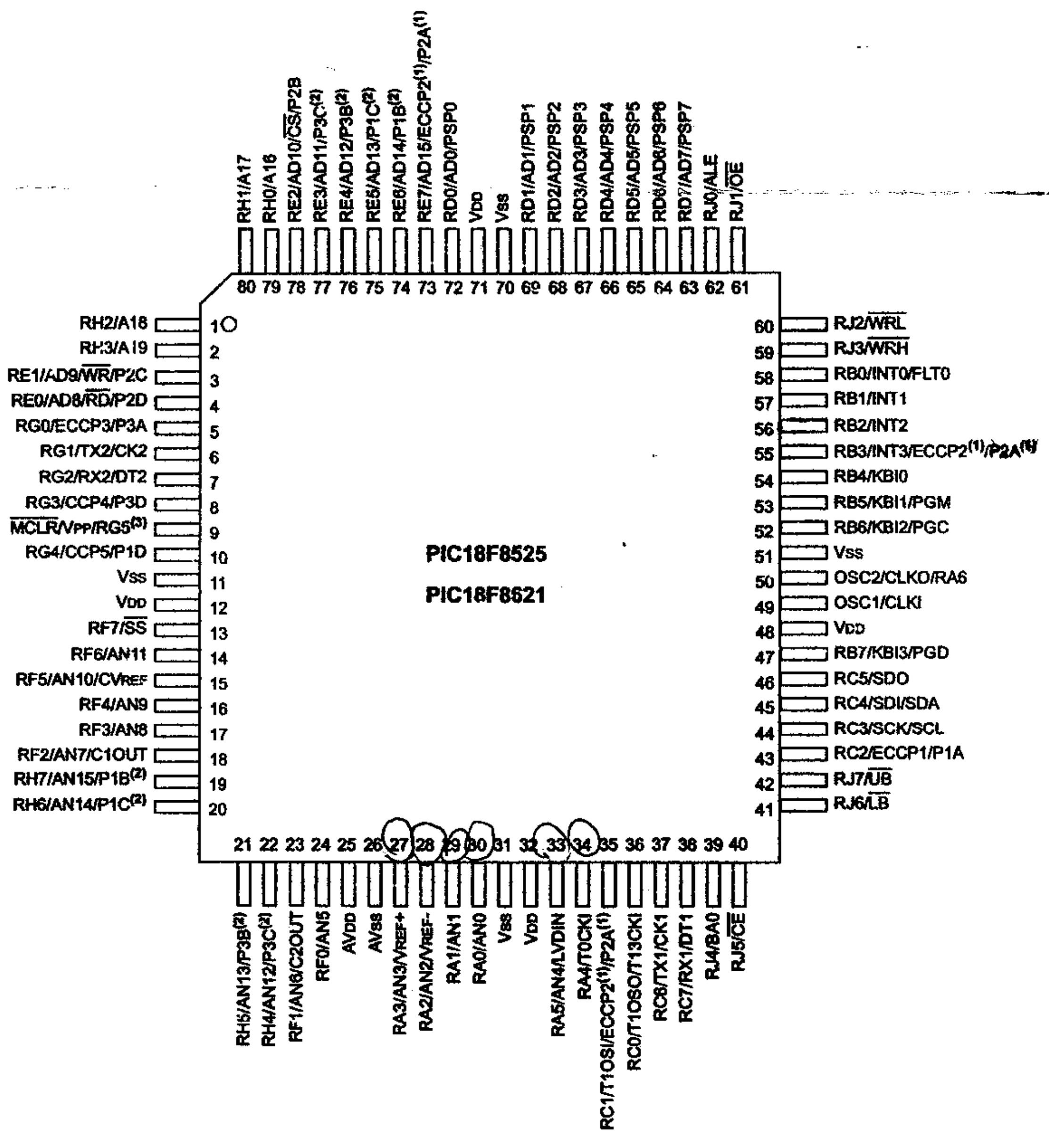

AirDrop-P
PIC18LF8621
CF CARD CONNECTOR
J4
U2
SP3232
SP3232 PIN 16 = Vcc
SP3232 PIN 15 = GND
SP3232 BYPASS CAP
RS-232 CONNECTOR
J2/DB9 FEMALE
RTSIN
RXIN
CTSOUT
TXOUT
+3.3VDC
+6-9VDC
J3
D2
1N5819
D1
MCLR
PGD
PGC
VR1
LM1086CS-3.3
IN
OUT
GND
R2
332
C2
C3
10uF
LED1
C1
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
.1uF
OSC2
1 - B
2 - O
3 - Br
4 - W/B
5 - W/O
6 - W/Br


+5V
+12V
SUPPLY
GND
IC1
VCC
VS
J1
J5MM
R1
R2
ENABLE_A
ENABLE_B
SEN_A
SEN_B
INPUT
INPUT1
INPUT2
INPUT3
INPUT4
OUT1
OUT2
OUT3
OUT4
FROM UCONTROLLER
L298
D2 1N4004
D3 1N4004
D4 1N4004
D5 1N4004
D6 1N4004
D7 1N4004
D8 1N4004
D9 1N4004
TO DC MOTOR
OUTPUT
H-BRIDGE
4/27/2006 06:56:08p
Sheet: 1/1


SUPPLY
OUTPUT
INPUT
D2
D3
D4
D5
D6
D7
D8
D9
1N4004
R1
R2

n Diagrams (Cont.'d)

80-Pin TQFP



Note 1: ECCP2/P2A are multiplexed with RC1 when CCP2MX is set; with RE7 when CCP2MX is cleared and the device is configured in Microcontroller mode; or with RB3 when CCP2MX is cleared in all other program memory modes.

2: P1B/P1C/P3B/P3C are multiplexed with RE6:RE3 when ECCPMX is set and with RH7:RH4 when ECCPMX is not set.

3: RG5 is multiplexed with MCLR and is only available when the MCLR Resets are disabled.

APPENDIX C

Source code: Remote Control System

```
Option Explicit

Dim KeyArray(100) As Boolean
Dim Calidad As Byte
Dim IncreaseCal As Boolean
Dim ReduceCal As Boolean
Dim intQuality As Integer

Private Sub ForwardCar()
   If optType(1).Value = True Then
   sock.SendData "Forward": DoEvents
   Else
   sock.SendData "AutoForward": DoEvents
   End If
End Sub

Private Sub ReverseCar()
   If optType(1).Value = True Then
   sock.SendData "Reverse": DoEvents
   Else
   sock.SendData "AutoReverse": DoEvents
   End If
End Sub

Private Sub LeftCar()
   If optType(1).Value = True Then
   sock.SendData "Left": DoEvents
   Else
   sock.SendData "AutoLeft": DoEvents
   End If
End Sub

Private Sub RightCar()
   If optType(1).Value = True Then
   sock.SendData "Right": DoEvents
   Else
   sock.SendData "AutoRight": DoEvents
   End If
End Sub

Private Sub btnFwd_Click()
   ForwardCar
End Sub

Private Sub btnRvs_Click()
   ReverseCar
End Sub

Private Sub btnLft_Click()
   LeftCar
End Sub

Private Sub btnRgt_Click()
   RightCar
End Sub

Private Sub btnStop_Click()
   sock.SendData "Stop": DoEvents
End Sub

Private Sub Video()
Dim Xs As Long, Ys As Long
      Xs = ImVideo.Width / Screen.TwipsPerPixelX
      Ys = ImVideo.Height / Screen.TwipsPerPixelY
   SetWindowRgn ImVideo.hwnd, CreateRoundRectRgn(0, 0, Xs, Ys, 12, 12), True

   PicMarco.Height = ImVideo.Height + 120
   PicMarco.Width = ImVideo.Width + 120

      Xs = PicMarco.Width / Screen.TwipsPerPixelX
      Ys = PicMarco.Height / Screen.TwipsPerPixelY
   SetWindowRgn PicMarco.hwnd, CreateRoundRectRgn(0, 0, Xs, Ys, 12, 12), True
End Sub

Private Sub CmdDisconnect_Click()
   sock.Close
'   LblCalidad.Caption = ""
'   LblFps.Caption = ""
   StatusBarCon.Panels(2).Text = ""
   StatusBarCon.Panels(3).Text = ""
   PicCalidad.Cls
   Set ImVideo.Picture = LoadPicture()
'   LblConVideo.Caption = " DISCONNECTED "
   StatusBarCon.Panels(1).Picture = PicDisConn
   StatusBarCon.Panels(1).Text = "Disconnected"
   CmdConnect.Enabled = True
   CmdDisconnect.Enabled = False
   btnStop.Enabled = False
   btnFwd.Enabled = False
   btnRvs.Enabled = False
   btnLft.Enabled = False
   btnRgt.Enabled = False
End Sub

Private Sub CmdConnect_Click()
On Error Resume Next
If TxtIp.Text = "" Or TxtPort.Text = "" Then
   MsgBox "Please specify the Port and IP address.", vbInformation, "Incomplete Information"
Else
   If sock.State <> sckClosed Then
   MsgBox "Please make sure the WiFi Car is on.", vbInformation, "General Information"
   Exit Sub
   End If
   sock.Connect TxtIp.Text, TxtPort.Text
End If
End Sub

Private Sub sock_Connect()

'   LblConVideo.Caption = "CONNECTED"
   StatusBarCon.Panels(1).Picture = PicConn
   StatusBarCon.Panels(1).Text = "Connected"
   Calidad = 2
   intQuality = Calidad
   ColorQuality (Calidad)
   ScrollQuality.Value = Calidad

   Do
      DoEvents
   Loop Until sock.State = sckConnected

   CmdConnect.Enabled = False
   CmdDisconnect.Enabled = True
   btnStop.Enabled = True
   btnFwd.Enabled = True
   btnRvs.Enabled = True
   btnLft.Enabled = True
   btnRgt.Enabled = True

End Sub
```

```
Private Sub sock_Close()
   sock.Close
   Set ImVideo.Picture = LoadPicture()

   'LblCalidad.Caption = ""
   StatusBarCon.Panels(2).Text = ""
   'LblFps.Caption = ""
   StatusBarCon.Panels(3).Text = ""
   PicCalidad.Cls
   PicCalidad.Refresh
'   LblConVideo.Caption = " DISCONNECTED "
   StatusBarCon.Panels(1).Picture = PicDisConn
   StatusBarCon.Panels(1).Text = "Disconnected"

   CmdConnect.Enabled = True
   CmdDisconnect.Enabled = False
   btnStop.Enabled = False
   btnFwd.Enabled = False
   btnRvs.Enabled = False
   btnLft.Enabled = False
   btnRgt.Enabled = False

End Sub

Private Sub sock_DataArrival(ByVal bytesTotal As Long)
Static StringDatosVi As String
Static HoraVi As String
Static Fps As Integer

Dim qVi As String
On Error Resume Next

      While (sock.BytesReceived > 0)
            sock.GetData qVi, vbString, bytesTotal
            StringDatosVi = StringDatosVi & qVi
      Wend

      'If Mid(StringDatosVi, Len(StringDatosVi) - 4, 5) = "Iiron" Then
      If Right(StringDatosVi, 5) = "Iiron" Then
         Dim Trozo As String
         Trozo = Mid(StringDatosVi, 2, Len(StringDatosVi) - 6)

                     If IncreaseCal = True Or ReduceCal = True Then
                        If IncreaseCal Then
                           IncreaseCal = False
                           If sock.State = 7 Then sock.SendData "Eliron": DoEvents
                        Else
                           ReduceCal = False
                           If sock.State = 7 Then sock.SendData "Fliron": DoEvents
                        End If
                     End If

                     StatusBarCon.Panels(2).Text = "Kpf : " & Format(Len(Trozo) / 1024, "#,##0.#0")
                     LblCalidad.Caption = "Kpf : " & Format(Len(Trozo) / 1024, "#,##0.#0")
                     LblCalidad.Refresh
                        Call LoadFromString(Trozo, ImVideo.hDC, 0, 0, (ImVideo.Width / 15), (ImVideo.Height / 15),
False)
                        ImVideo.Refresh


                     If HoraVi = "" Then HoraVi = Format(Now, "ss")
                        If HoraVi = Format(Now, "ss") Then
                           Fps = Fps + 1
                        Else
                           StatusBarCon.Panels(3).Text = "Fps : " & Fps
                           LblFps.Caption = "Fps : " & Fps
                           LblFps.Refresh
```

```
                  Fps = 0
                  HoraVi = Format(Now, "ss")
                End If

        StringDatosVi = ""

    End If
End Sub


Private Sub Form_Unload(Cancel As Integer)
  If sock.State = sckConnected Then
    Call sock_Close
  End If

  Do
    DoEvents
  Loop Until sock.State = sckClosed
End Sub

Private Sub CmdExit_Click()
  If sock.State = sckConnected Then
    Call sock_Close
  Else
    Call sock_Close
  End If

  Do
    DoEvents
  Loop Until sock.State = sckClosed

  Unload Me
End Sub


Private Sub ScrollQuality_Change()
If sock.State = 7 Then
  If ScrollQuality.Value > intQuality Then
    If Calidad < 17 Then
      'CmdCalidad(0).Enabled = False
      PicCalidad.Cls
      Calidad = ScrollQuality.Value
      ColorQuality (Calidad)
      IncreaseCal = True
    End If
  Else
    If Calidad > 1 Then
      'CmdCalidad(1).Enabled = False
      PicCalidad.Cls
      Calidad = ScrollQuality.Value
      ColorQuality (Calidad)
      ReduceCal = True
    End If
  End If
End If
intQuality = Calidad
End Sub

Private Sub ColorQuality(quality As Integer)
  If quality < 3 Then
    PicCalidad.Line (0, 0)-(Calidad * 255, 120), vbRed, BF
  ElseIf quality >= 3 And quality < 13 Then
    PicCalidad.Line (0, 0)-(3 * 255, 120), vbRed, BF
    PicCalidad.Line (3 * 255, 0)-(Calidad * 255, 120), vbYellow, BF
  Else
    PicCalidad.Line (0, 0)-(3 * 255, 120), vbRed, BF
    PicCalidad.Line (3 * 255, 0)-(12 * 255, 120), vbYellow, BF
```

```
        PicCalidad.Line (12 * 255, 0)-(Calidad * 255, 120), vbGreen, BF
    End If
End Sub



Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
On Error Resume Next
    'Markes the element in the array as true if it the key is pushed
    KeyArray(KeyCode) = True
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
On Error Resume Next
    'Markes the element in the array as false if it the key is released
    KeyArray(KeyCode) = False
    btnStop.Enabled = True
    btnFwd.Enabled = True
    btnRvs.Enabled = True
    btnLft.Enabled = True
    btnRgt.Enabled = True
End Sub
'
Private Sub Form_Load()

    'Setting the interval of the "game loop"/timer to 10 for the Array version
    With Me
        .KeyPreview = True
    End With
    tmrArray.Interval = 10

    CmdConnect.Enabled = True
    CmdDisconnect.Enabled = False
    Call Video
End Sub

Private Sub tmrArray_Timer()
    If sock.State = 7 Then
    If KeyArray(39) = True Then     'moves right
        RightCar
        btnRgt.Enabled = False
    End If
    If KeyArray(37) = True Then     'moves left
        LeftCar
        btnLft.Enabled = False
    End If
    If KeyArray(38) = True Then     'moves up
        ForwardCar
        btnFwd.Enabled = False
    End If
    If KeyArray(40) = True Then     'moves down
        ReverseCar
        btnRvs.Enabled = False
    End If
    If KeyArray(83) = True Then     'moves down
        sock.SendData "Stop": DoEvents
        btnStop.Enabled = False
    End If
    End If
End Sub
```

Source Code: Wi-Fi Car System

```
Option Explicit

Dim Terminado As Boolean
Dim KeyArray(100) As Boolean
Private lngCountDownValue As Long '// variable that stores countdown value


Private Sub tmrArray_Timer()
   If KeyArray(39) = True Then     'moves right
      RightCar
      btnRgt.Enabled = False
   End If
   If KeyArray(37) = True Then     'moves left
      LeftCar
      btnLft.Enabled = False
   End If
   If KeyArray(38) = True Then     'moves up
      ForwardCar
      btnFwd.Enabled = False
   End If
   If KeyArray(40) = True Then     'moves down
      ReverseCar
      btnRvs.Enabled = False
   End If
   If KeyArray(83) = True Then     'stop
      StopCar
      btnStop.Enabled = False
   End If
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
On Error Resume Next
   'Markes the element in the array as true if it the key is pushed
   Picture1.TabIndex = 0
   KeyArray(KeyCode) = True
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
On Error Resume Next
   'Markes the element in the array as false if it the key is released
   KeyArray(KeyCode) = False
   btnStop.Enabled = True
   btnFwd.Enabled = True
   btnRvs.Enabled = True
   btnLft.Enabled = True
   btnRgt.Enabled = True
End Sub

Public Sub Enviar(Linea As String)
   If sock.State = sckConnected Then sock.SendData Linea: DoEvents Else Bsent = False
End Sub

Private Sub tmrWait_Timer()
   lngCountDownValue = lngCountDownValue - 1 '// decrement the value
   If lngCountDownValue = 0 Then
      tmrWait.Enabled = False
      StopCar
   End If
End Sub

Private Sub delay()
   lngCountDownValue = 2 '// count down from 10
   tmrWait.Interval = 1000 '// 1 second
   Call tmrWait_Timer
   tmrWait.Enabled = True '// timer is enabled
End Sub

rivate Sub ForwardCar()
   Call PortOut(888, 6)
   lstMsg.AddItem ("Forward")
   If optType(1).Value = True Then
   delay
   End If
nd Sub

rivate Sub LeftCar()
   Call PortOut(888, 4)
   lstMsg.AddItem ("Left")
   If optType(1).Value = True Then
   delay
   End If
nd Sub

rivate Sub RightCar()
   Call PortOut(888, 2)
   lstMsg.AddItem ("Right")
   If optType(1).Value = True Then
   delay
   End If
nd Sub

rivate Sub ReverseCar()
   Call PortOut(888, 9)
   lstMsg.AddItem ("Reverse")
   If optType(1).Value = True Then
   delay
   End If
nd Sub

rivate Sub StopCar()
   Call PortOut(888, 16)
   lstMsg.AddItem ("Stop")
nd Sub

rivate Sub sock_SendComplete()
   Bsent = True
nd Sub

rivate Sub btnFwd_Click()
   ForwardCar
nd Sub

rivate Sub btnRvs_Click()
  ReverseCar
nd Sub

rivate Sub btnLft_Click()
   LeftCar
nd Sub

rivate Sub btnRgt_Click()
   RightCar
nd Sub

rivate Sub btnStop_Click()
   StopCar
nd Sub

rivate Sub call_con()
f sock.State <> sckClosed Then Exit Sub
   sock.Close
  sock.Bind CInt(TxtPort.Text), TxtIp.Text
  sock.Listen
  StatusBarCon.Panels(1).Text = "Listening... "
  StatusBarCon.Panels(1).Picture = PicListen
  Bsent = True
  capSetCallbackOnFrame lwndC, AddressOf MyFrameCallback
nd Sub

rivate Sub sock_ConnectionRequest(ByVal requestID As Long)
  sock.Close
  sock.Accept requestID
  Calidad = 14
   MsgBox "Connected with " & sock.RemoteHostIP
  StatusBarCon.Panels(1).Text = "Connected"
  StatusBarCon.Panels(1).Picture = PicConn
  Bsent = True
  capSetCallbackOnFrame lwndC, AddressOf MyFrameCallback
nd Sub

rivate Sub CmdDisconnect_Click()
  Call sock_Close
nd Sub


rivate Sub CmdQuit_Click()
  Cerrando = True
  Call PortOut(888, 0)
  If sock.State = sckConnected Then
     Call sock_Close
  End If

  Do
     DoEvents
  Loop Until EnCallback = False

  Call Form_Unload(0): Unload Me

nd Sub



rivate Sub sock_Close()
  sock.Close
  StatusBarCon.Panels(1).Text = "Disconnected "
  StatusBarCon.Panels(1).Picture = PicDisconn
  StatusBarCon.Panels(2).Text = ""
  StatusBarCon.Panels(3).Text = ""
  lstMsg.Clear

  If (lGDIP) Then
     GdiplusShutdown lGDIP
  End If

  call_con
nd Sub

rivate Sub sock_Connect()
  StatusBarCon.Panels(1).Text = "Connected "
  StatusBarCon.Panels(1).Picture = PicConn
  Calidad = 75
  Do
     DoEvents
  Loop Until sock.State = sckConnected
  Bsent = True
nd Sub

rivate Sub sock_DataArrival(ByVal bytesTotal As Long)
```

```
Static StringDatos As String
Dim q As String, Inicio As String


    While (sock.BytesReceived > 0)
        sock.GetData q, vbString, bytesTotal
        StringDatos = StringDatos & q
    Wend

    'If Mid(StringDatos, Len(StringDatos) - 4, 5) = "liron" Then

    If Right(StringDatos, 5) = "liron" Then

            Inicio = Mid(StringDatos, 1, 1)
              Select Case Inicio
                Case "E"
                  StringDatos = ""
                  If Calidad + 5 <= 90 Then Calidad = Calidad + 5
                Case "F"
                  StringDatos = ""
                  If Calidad - 5 >= 10 Then Calidad = Calidad - 5
                Case "T" 'detectar Movimiento
                  If DetectarMovimiento = False Then DetectarMovimiento = True Else DetectarMovimiento = False
              End Select
    ElseIf StringDatos = "Forward" Then
    optType(1).Value = True
    ForwardCar
    ElseIf StringDatos = "Reverse" Then
    optType(1).Value = True
    ReverseCar
    ElseIf StringDatos = "Left" Then
    optType(1).Value = True
    LeftCar
    ElseIf StringDatos = "Right" Then
    optType(1).Value = True
    RightCar
    ElseIf StringDatos = "Stop" Then
    StopCar
    ElseIf StringDatos = "AutoForward" Then
    optType(0).Value = True
    ForwardCar
    ElseIf StringDatos = "AutoReverse" Then
    optType(0).Value = True
    ReverseCar
    ElseIf StringDatos = "AutoLeft" Then
    optType(0).Value = True
    LeftCar
    ElseIf StringDatos = "AutoRight" Then
    optType(0).Value = True
    RightCar
    End If
    StringDatos = ""
End Sub

Private Sub Form_Load()

  Dim lpszName As String * 100
  Dim lpszVer As String * 100
  Dim Caps As CAPDRIVERCAPS

  StatusBarCon.Panels(1).Picture = PicDisconn
  StatusBarCon.Panels(1).Text = "Disconnected"
  lstMsg.Clear

  StopCar
  Calidad = 75
```

```
    '//Create Capture Window
    capGetDriverDescriptionA 0, lpszName, 100, lpszVer, 100  '// Retrieves driver info
    lwndC = capCreateCaptureWindowA(lpszName, WS_VISIBLE Or WS_CHILD, 5, 5, 160, 120, Me.hWnd, 0)

    '// Set title of window to name of driver
    SetWindowText lwndC, lpszName

    '// Set the video stream callback function
    capSetCallbackOnStatus lwndC, AddressOf MyStatusCallback
    capSetCallbackOnError lwndC, AddressOf MyErrorCallback

    '// Connect the capture window to the driver
    If capDriverConnect(lwndC, 0) Then
        '/////
        '// Only do the following if the connect was successful.
        '// if it fails, the error will be reported in the call
        '// back function.
        '/////
        '// Get the capabilities of the capture driver
        capDriverGetCaps lwndC, VarPtr(Caps), Len(Caps)

        '// If the capture driver does not support a dialog, grey it out
        '// in the menu bar.
'        If Caps.fHasDlgVideoSource = 0 Then mnuSource.Enabled = False
'        If Caps.fHasDlgVideoFormat = 0 Then mnuFormat.Enabled = False
'        If Caps.fHasDlgVideoDisplay = 0 Then mnuDisplay.Enabled = False

        '// Turn Scale on
        capPreviewScale lwndC, True

        '// Set the preview rate in milliseconds
        capPreviewRate lwndC, 66

        '// Start previewing the image from the camera
        capPreview lwndC, True

        '// Resize the capture window to show the whole image
        ResizeCaptureWindow lwndC

        SetWindowPos Me.hWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOSIZE Or SWP_NOMOVE
    call_con
    With Me
        .KeyPreview = True
    End With
    tmrArray.Interval = 10
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If sock.State = sckConnected Then
        Call sock_Close
    End If

    If EnCallback = False Then capSetCallbackOnFrame lwndC, vbNull

End Sub


Private Sub Form_Unload(Cancel As Integer)
        '// Disable all callbacks
        capSetCallbackOnError lwndC, vbNull
        capSetCallbackOnStatus lwndC, vbNull
        capSetCallbackOnYield lwndC, vbNull
        If EnCallback = False Then capSetCallbackOnFrame lwndC, vbNull
        capSetCallbackOnVideoStream lwndC, vbNull
        capSetCallbackOnWaveStream lwndC, vbNull
        capSetCallbackOnCapControl lwndC, vbNull
End Sub
```

APPENDIX D

Message: 1-60 airdrop_user

I'm experiencing an issue with the rd_xxx routines I wonder if anyone else has. I've found that there is a delay required after the controlling pin (OE in the code snippet below). The 2ms I've included below is an empirical number, but seems to work in the gnu-avr compiler (I'm planning on moving over to CrossWorksAVR as soon as I get one small issue resolved). I'm not sure what Intersil's spec requires, but this seems to make my port reliable. comments?

Thanks,

Dave

```
unsigned char rd_cf_reg( unsigned int reg_addr )
{
        unsigned char data;
        unsigned char i;
        wr_cf_addr( reg_addr );
        clr_OE;
        delay_ms(2);        <========
        i=1;
        while(--i);
        data = data_in;
        set_OE;
        NOP();
        return(data);
}
```

Hi All,

Well, since receiveing my AirDrop-A I've been working on code to put into it. :-) I'm currently working on a uIP port (which is *almost* there) and Ethereal is your friend (when the packets get out...)
I've customized uIP so that it can use an 802.11 link layer and am liasing with Adam Dunkels, the author of uIP, on how best to present the stuff. I also wrote a new BASIC interpreter over the Christmas break, called NetBASIC, that uses uIP to provide a telnet server so you can telnet to the interpreter and start bashing away programming the thing.
This is fully functional on one of our MSP430F1611 board with a CS8900A Ethernet chip, and I'm now moving it to the AirDrop-A using our C compiler, uIP, and the skeleton code provided by Fred.

I'll let you know how I get on.

Hi Dave,
Hmm, I haven't needed to add that, mine seems to work OK, but I *did* need to increase the timeout. CrossWorks for AVR generates tighter loop code than ICCAVR, so I needed to attend to the issue. As such, I've increased all the delays by a factor of 3 to get the AirDrop-A reliable:

```
char rd_cf_reg(unsigned int reg_addr)
{
        char data,i;
        wr_cf_addr(reg_addr);
        clr_OE;
        i=3;  // ICC=1
        while(--i);
        data = data_in;
        set_OE;
        NOP();
        return(data);
}
```

I just modified the code throughout. It took a bit of time to get going and I did ask Fred about the delay timing--he said it was empirical and he got a scope out and looked at it to tune the delays. However, if we're all using different versions of ICCAVR, or different compilers such as CrossWorks for AVR, those pre-programmed delays will indeed be problematic. It's probably either (a) best to go real conservative or (b) use a compiler intrinsic such as __delay_cycles in CrossWorks that will plumb in the exact delays.
Regards,

Hi All, Fred, Peter,

When I ping an AirDrop-A running stock firmware and use Ethereal to dump the packets on the line, I get this type of behaviour for four ping packets:

```
Echo (ping) request
Echo (ping) request
Echo (ping) reply
Echo (ping) reply
Echo (ping) request
Echo (ping) request
Echo (ping) reply
Echo (ping) reply
Echo (ping) request
Echo (ping) request
Echo (ping) reply
Echo (ping) reply
Echo (ping) request
Echo (ping) request
Echo (ping) reply
Echo (ping) reply
```

When I ping another wireless device on my network I get what I expect to get:

```
Echo (ping) request
Echo (ping) reply
Echo (ping) request
```

1

Hence, I reckon the first packet turnaround is delayed because of the router or TRENDnet card setting up internal routing--what about an ARP cache?

By way of comparison, here's what I get using the orignal ImageCraft Image (not patched) Note that the first ping is much longer and could easily be explained as the access ping ARP caching.

First Ping:

```
Pinging 192.168.0.151 with 32 bytes of data:

Reply from 192.168.0.151: bytes=32 time=277ms TTL=123
Reply from 192.168.0.151: bytes=32 time=166ms TTL=128
Reply from 192.168.0.151: bytes=32 time=166ms TTL=128
Reply from 192.168.0.151: bytes=32 time=167ms TTL=126

Ping statistics for 192.168.0.151:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 166ms, Maximum = 277ms, Average = 194ms
```

2nd and subsequent pings:

```
Reply from 192.168.0.151: bytes=32 time=167ms TTL=128
Reply from 192.168.0.151: bytes=32 time=167ms TTL=128
Reply from 192.168.0.151: bytes=32 time=166ms TTL=128
Reply from 192.168.0.151: bytes=32 time=166ms TTL=128
```

The test network is a sole access point (Belkin) with no other nodes. I have 2 access points. Interesting the difference in Paul's response verses this one. Patch?

Thanks Julian. I thing what I'm going to do is put the scope on a data line and I/O pins and characterize what the issue is. I'm faily positive there is an access time issue here. Based on what I see, comparison at the assembly level between the various compiler soce would be really interesting. I'll let you know what I see.

Dave

Greetings,

[This is my first post. If my note is not useful or helpful, please advise and I will refrain from further postings like this. Thank you]

I'm curious about the timing of these bus cycles also. I'm no expert by any means. But it seems to me that:

1) some timing would be defined by the Compact Flash spec itself
2) the rest would be the Prism chip.

You can freely download the Compact Flash spec from here:

http:[illegible]compactflash.org/specdl1.htm

One card that looks interesting, and has a very good manual, including timing diagrams for the bus cycles, is the Sandisk Connectplus. However, it is a little wierd, since it has both a radio and 128MB of flash on the same card.

Here is the manual:
[illegible]sandisk.com/pdf/industrial/ProdManualCFWLANv1.0.pdf

Now about the prism chipset. I've been trying to locate all of the information that is available without an NDA.

I am guessing that for Compact Flash cards, all of them are using the "prism 3" chipset. Does anyone know if there is a specific chip for compact flash, or is it actually using the "pcmcia" chip, the ISL3871.

Another possibility might be "prism 2.5" chipset. I think the chip of interest in that chipset would be the ISL3873B.

From a programmer's point of view, the Linux drivers seem to imply that the prism "2, 2.5 and 3" chips are all about the same. If so, then you may want to look into the data available for the older HFA3842 chip.

I have found the open source "hostap" Linux driver to be pretty useful. It at least has "#defines" for the prism registers. You can read about it here:

http://hostap.epitest.fi/
Hope this information is useful to you all.

William

This is really going to show my lack of knowledge w.r.t. the compact/FLASH, but my assumptions are that the compact FLASH spec would be a recommended interface timing spec (i.e. a bus spec) where the spec. for the Prism chip would, of course, be the implementation of the particuler interface. You raise a good point; That is, is the what are the specifics of the interface and where the timing fits in to that spec. verses the implementation timing. I'll download the documents you referenced and have a look-see.

> I am guessing that for Compact Flash cards, all of them are using the "prism 3" chipset. Does anyone know if there is a specific chip for compact flash, or is it actually using the "pcmcia" chip, the ISL3871. Another possibility might be "prism 2.5" chipset. I think the chip of interest in that chipset would be the ISL3873B.

Fred (or Peter) is is possibly to mention which flavor of the Prisim is in the TEW-222CF without dishonoring the NDA you have with Trident?

I am running my 802.11g network with a couple of 802.11b devices (Tosh laptop, AirDrop). All other nodes are g nodes. I have a Windows Media Server running a MusicMatch 10 UPnP server to four Netgear MP101s (shite boxes, don't ever purchase one). My router is a Linksys WRT54G. I'm very happy with the router...

```
The patch the Fred advised was:

void
```

(3)

```
// Send

TxF = get_free_TxF();
if (TxF == 0)
  rc = 1;
else
  {
    if (uip_len < 40 + UIP_LLH_LEN)
      b_write_3(TxF, rxdatalength_offset,
                &len, 2,
                uip_buf, uip_len,
                0, 0);
    else
      b_write_3(TxF, rxdatalength_offset,
                &len, 2,
                uip_buf, UIP_LLH_LEN+40,
                uip_appdata, uip_len - 40 - UIP_LLH_LEN);

    if (rc = send_command(0x010B, TxF))
      {
        ;
        //printf("\r\nTransmit failed");
      }
    do_event_housekeeping();
  }
}
```

uip_buf contains the variable part of the frame, and uip_appdata can come from "somewhere else". If you really need to have a 1500-byte variable outgoing packet, you need to declare that uip_buf is of that size but for many applications this is not necessary.

The 0x10B is a TX command with the RECYCLE bit set to recycle the buffer
when finished.

Hi All,

I just wondered what other people are going to use their AirDrops for? It is general tinkering or do you have real uses? Tinkering is OK (I do
lots of it) but knowing about some real use for these things is always entertaining and interesting.

My interest is that I want to get NetBASIC running on the AVR. I already have it running on some larger Ethernet-based MSP430 boards and this is the first port to an AVR. Not one to change one thing at a time, I went the whole hog and changed architecture (MSP430 to AVR), compiler (CrossWorks for MSP430 to CrossWorks for AVR) and link layer (CS8900A LAN to PRISM-based WLAN). It's taken me a while to get all mu ducks in a line, but at least I'm there now. :-) You gotta use your product in anger to know what it is good and bad at, and where the wrinkles are. The last week has been enligtening.

The AirDrop has been frustrating (getting packets out of the sucker, Ethereal traces, you name it) and has, of course, given that rush moment
when things start to work and the NetBASIC signon came up on the screen.
It's been 3am and 4am before I went to bed for the whole of this week...

Now, I just need an AirDrop with a stonking 32K RAM chip on it and that would make NetBASIC just so cute on the AVR...

Regards,

My initial use is that I am playing with robots, and I want to do rather more than the old boring 'follow a line' or 'don't bump into things'. To do good stuff with navigation, map-making, etc I need full-blown PC backup. So, I need to allow my Robot to interface to a PC (running a suite of JINI/JAVA apps). The link had to be wireless, so that got it down to wireless RS232, Bluetooth and 802.11b. 802.11b won because:

(a) It's faster

(b) I don't need an adapter at the other end, I can just make the robot a device on my network

(c) It hugely simplifies the Java programming

Oh yes, and I'm also looking at working the opposite direction to Paul and pushing the AirDrop code onto a MSP430.
That's just tinkering!

Julian

Argh! Perhaps I'll stick with Fred's code for a bit longer and just poach bits of TCP as and when .
Over the short term I'm trying to send high bitrate audio from the airdrop-p to a pc. also, I want to statistically characterize packet errors and losses at the pc side, and possibly have it talk back to

7

The symptoms are that I can apparently load a program under AVR Studio and I can set the fuse bits, but unfortunately I can't read any registers or any memory out of the AVR. Because of this, I don't know whether it is programmed or not. I've tried setting the fuse bits to clock internally and externally and neither allows me to debug the AVR (although the OCD fuse is enabling OCD debug).

My current fuse bits are 0xff 0x19 0xdf which is ok, I think. Lock bits are 0xff. The AirDrop-A just suddenly stopped working...

Any ideas?

No matter, a lot of blind fiddling fixed it. Strange.

Hi Bill,

Well, you can do exactly the same on Windows:

Ping -l 1462 192.168.0.110 using stock AirDrop-A firmware on my WLAN:

```
Pinging 192.168.0.110 with 1462 bytes of data:

Reply from 192.168.0.110: bytes=1462 time=53ms TTL=128
Reply from 192.168.0.110: bytes=1462 time=53ms TTL=128
Reply from 192.168.0.110: bytes=1462 time=55ms TTL=128
Reply from 192.168.0.110: bytes=1462 time=53ms TTL=128
```

Using my modified firmware which underpins my uIP port:

```
Pinging 192.168.0.110 with 1462 bytes of data:

Reply from 192.168.0.110: bytes=1462 time=47ms TTL=128
Reply from 192.168.0.110: bytes=1462 time=47ms TTL=128
Reply from 192.168.0.110: bytes=1462 time=47ms TTL=128
Reply from 192.168.0.110: bytes=1462 time=47ms TTL=128
```

The modified firmware is, uh, modified. YMMV.

* Peter

Hi Guys,

I now can report that I have uIP working on the AirDrop-A! Yes, I can telnet into my NetBASIC interpreter running on the AirDrop, write (small) programs in BASIC and run them wirelessly using a telnet session. :-) uIP in this configuration is about 5K, with telnet services and ARP. You can add in an HTTP server too, and it gets alittle bigger (mostly the filing system).
When I get it a little more stable, I'll try and post some results.

Rgds,

Hi, Paul
Thanks for pointing that out...And , thanks for the fix :-) I would have been more than happy to publish the defines.

Unfortunately, without the NDA with Conexant none of this may have happened :-) So, I decided to play the game and hope for the best. The good news is that I see all of you "breaking the code" by using
my somewhat cryptic source (an understatement) as a baseline and associating it with open source documentation. As far as I know, there is no law against that. So, keep it up and all of my code will begin to make lots of sense to you all. The whole idea behind this project was to give us embedded guys a way to use 802.11b to our advantage :-) I am thrilled with what I am seeing come out of this group!

Greetings,
Is anyone running a bootloader on your airdrop-P ?
If not, do you re-flash the part via LVP or HVP?
My homebrew ICSP is high-voltage only.
I'm considering a "mod" to the board-- lift the output leg of the 3.3 volt regulator, and add a "jumper" to the prototyping area and then I could run the board at 5 volts while doing ICSP.
Any other suggestions?
Thanks,

William

Paul,

Great news. I'm currently working on encapsulating the CF drivers in the PIC version, so they're no longer mixed in with the TCP / ICMP / IP / ARP code, and you have a basic

getFrame()

putFrame()

waitForData()

waitToSend()

initNIC()

type interface. Will doing this put me in a good place to port in uIP? (I haven't looked at it thoroughly yet, as I am waiting for a MSP IP board which is currently waiting for me to pay the customs charge!)

Julian

You know, I reckon it does customers and businesses alike a disservice when they hide their technical details behind an NDA. The "join the club" mentality really makes me prickly.

(5)

```
{
        unsigned char defSSID[]="";


        airdrop_config2(BSS,0,NULL);
        airdrop_config2(SSID,0,defSSID);
        if(key!=NULL)
        {
                if(strlen((char *)key)==5) airdrop_config2(WEP_SET_KEY_64,0,key);
                else airdrop_config2(WEP_SET_KEY_128,0,key);
                airdrop_config2(WEP_USE_KEY,0,NULL);
                airdrop_config2(WEP_AUTH_PRIVATE,0,NULL);
                airdrop_config2(WEP_ON,0,NULL);
        }
        else airdrop_config2(WEP_OFF,0,NULL);
}


void main(void)
{
        unsigned int i,temp,evstat_data;
        unsigned char rc;


        unsigned char preferredSSID[]="";
        unsigned char WEPKey0[]={0x51,0x79,0x36,0x0e,0x4c,0};


        init_USART(57600);
        init_cf_card();
```

```
        //airdrop_config2(BSS,0,NULL);
        //airdrop_config2(SSID,0,preferredSSID);
        //airdrop_config2(WEP_SET_KEY_64,0,WEPKey0);
        //airdrop_config2(WEP_USE_KEY,0,NULL);
        //airdrop_config2(WEP_AUTH_PRIVATE,0,NULL);
        //airdrop_config2(WEP_ON,0,NULL);


        wifi_init(WEPKey0);


        ....
```

If you change the last line to wifi_init(NULL) you'll see that wifi_init selects the WEP_OFF option from airdrop_config2. For a 64-bit key, substitute your key values in place of the 5 non-zero bytes in WEPKey0[]. For a 128-bit key, initialise WEPKey0[] to contain the 13 bytes of key data followed by a zero byte.

To explain:

BSS / IBSS : selects whether this is an infrastructure of ad-hoc network

SSID : sets the preferred SSID – "" means connect to any

WEP_SET_KEY_* : sets the specified key (0, 1, 2, 3) with a 64-bit or 128-bit key

WEP_USE_KEY : tells the CF card which key (0, 1, 2, 3) to use

WEP_AUTH_* : chooses the authentication mode – shared or private

WEP_ON / WEP_OFF : pretty obvious

I'm aware that the (rather crude) mechanism of treating WEPKey0[] as a string, using a 0 byte to spot the end means you're limited to keys with non-zero final byte, but that's pretty minor, and you can always get round it by rewriting wifi_init to accept a keyLength parameter.

Julian

Paul,

Oh well. I should be getting a wired Ethernet MSP board tomorrow, so I'll start to read up on uIP.

--- airdrop_user : 2

Subject: Re: Uses for AirDrop? avrunit

Hi, Alex

I was intrigued by your alias "X".. Reminds me of the ZZ Top song "I heard it on the X" }

You are among friends... As Paul alluded to earlier, this secret crap concerning the PRISM chipset has got to be overcome..

In the very beginning of my interest in this, I went to the defacto standard Linux PRISM guys 30 miles south of me asking (begging) for a clue..

They said they couldn't help me because they had signed the NDA and needed to protect themselves, while at the same instant they were spouting out all kinds of open source PRISM-based Linux source code..

Go figure????????

Alex, you are here and therefore you too are hardcore....and you are welcomed..

Fred

Hi,

I'll ask the TRENDnet folks about that and see if we can get an answer..

In the meantime, you may be able to determine that by dumping the tuples in the CIS scan.. I am going to do that in the book, which I'm frantically

writing on now..(due date is end of Feb)..So, if someone beats me to it, I won't be too upset :-) However, I'll try to get that into my task list and post the results for everyone.

I'm under the assumption that all of the PRISM chipsets from 2 up to 3 are "identical" in operation...

I had a real hard time trying to use the compactflash specs for timing.. Their times seemed very short compared to what

I had to do to get the TRENDnet card to respond.. I basically made the card crawl and work

And then started shortening up the delays until the card quit working...

agreement could they have signed? arent the open source drivers [illegible]

for sure, it seems odd.

later

-Alex

PS: ZZTOP ROCKS

I have tried the Linksys cf-card WCF12 in the air-Drop and there seems to be no difference between it and the trend.

Maybe there is a clue to the chipset...

Manfred

--- airdrop_user : 2

Hi, Manfred

You can also use the yellow NetGear CF card and the bulging antennaed D-Link CF card..

Fred

They both use PRISM v2 or later .

Hi Fred,

> I'm under the assumption that all of the PRISM chipsets from 2 up to 3 are "identical" in operation...

> I had a real hard time trying to use the compactflash specs for timing..

> Their times seemed very short compared to what I had to do to get the TRENDnet card to respond.. I basically made the card crawl and work and then started shortening up the delays until the card quit working...

As with MMC, there's the spec and there's whaat's implemented. And invariably, the people writing the software to read the MMC cards have to assume that the card can be a bit slooooowwwww. There is a vast range of differences between MMC cards, not so much on SD cards. If you're talking to anything other than a piece of memory on the end of a CF card. I reckon it's all going to be device dependent.

However, those numbers for the delays, gee, *compiler* dependent. Have you got it working on CVAVR, Fred?

I'm going to try some throughput numbers to see what I can achieve, but right now I'm back on wired Ethernet for a while.

Rgds,
-- Paul.

Topic: Airdrop and QoS

Hi Fred, All,

I was going to port the uIP to Airdrop-A, but then I saw your [illegible]

Regards,

-martin

Hi,

I'm currently liasing with Adam Dunkels on this one. The problem is that uIP 0.9 is rather behind the times, the uIP in Contiki is better (faster, fewer bugs). I fixed some of the bugs in 0.9 but it's not being put back into the 0.9 release as I suspect Adam is doing other things. I modded uIP to take account of the LLC and SNAP headers, which was pretty easy.

I'm not tidying up the device driver by looking at the Linux code. When it's all finished, I'll post it. But right now, it's not really what you'd call ultra stable, probably because I'm pushing my luck a bit by engineering in quite a lot of stuff such as my BASIC interpreter, which leaves precious little RAM free.

Give me a week or so and I should have DHCP running reliably. It's all "beta quality" right now.

-Paul

Hi,

I am new to this group. I saw that there were many messages surrouding PRISM2 chipset. I am no expert in this, but I willing to share something I know:

1) There seems to be a PRISM2 programmer's manual floating around in the net.
http://home.eunet.cz/jt/wifi/RM0251.pdf

2) Roy Franz published a Circuit Cellar article about WiFi and microcontroller, he also published his driver. His driver is clean, but is not full-featured, and has some hacking he borrowed from the Linux driver. The article is Circuit Cellar 2003, #157. You can purchase the article for $1.00, and download the driver for free:
ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2003/157/Franz-157.zip

3) The Ethernut guys had done a WiFi project, they wrote a WiFi driver based on FreeBSD code. This driver seems to be feature complete (it has WEP, channel change, etc). The only difference(from Airdrop) is that they are not using GPIO to do read/write, but rather used a CPLD-base PCMCIA interface. You can download
everything from http://www.ethernut.de/en/wlan/

I am not sure if sharing this information is legal, everything above is found by doing a simple Google search. It's up to you to decide whether to download them or not.

All I am hoping is that someone (from this elite group) can unify all this, and create a working, full-featured, better-documented, microcontroller-based driver for Airdrop-A.

Thanks
-martin

---

just wondering if anyone has tested how much data throughput the port i/o to the cf card can handle...

i hope its not exceedingly slow, i had plans for minimum 22->44KBytes/second.

-alex

Hi,

→ I/O to the CF card is exceedingly quick. The question is one of TCP implementation, <u>though</u>. If you don't know a lot about TCP, you're going to be out of luck.

For instance, consider transmitting to the EDTP module from a PC. Now, on a wired MSP430 (8MHz) I can transmit 1,000 packets of 400 bytes in 6s, so that's ~66K/second and is limited by the MSP430 end. On an AVR over 802.11b it will be less, but I haven't tested that yet.

Going in the other direction is pure pain. Have you heard of delayed ACKs? To overcome the delayed ACK problem you need a fairly intelligent and <u>RAM-hungry TCP implementation</u>. Let's assume we're sending from AirDrop to PC and we're doing the sending with simple software. We send a packet from the AirDrop to the PC and we expect and ACK back--we wait for the ACK to arrive before sending the next packet. We'll try it our right now... Hey, wow! That's really slow! What's up? This is delayed ACK syndrome where the ACK from the PC is *delayed* on the assumption that another packet will be coming to the PC fairly rapidly and that the PC can ACK both packets on arrival of the second packet thus saving one ACK transmission.

Only trouble is, there *isn't* a second packet coming quickly after the first because the AirDrop is waiting for the PC's ACK and the PC is holding off the ACK waiting for the AirDrop to send it another packet!

So, you need at least two TCP segments in flight at any one time from the AirDrop. Ok, seems fair enough. Only for RAM-constrained programs is a bloody nightmare. Consider that you want to boost throughput by sending big packets of, say, 1500 bytes (inc headers). Delayed ACKs mean that you need to send two of these, but the rules of TCP mean that these two packets may get lost and eaten by a dog and require a retransmit. So, you need *two* 1500 byte packets hanging around. Bugger. That's 3K! I only have 1K left for something useful on the EDTP... Of course, you can reduce the MSS but that cuts outright performance *and* reduces the packet size you can receive. All around misery. Now, it would be *nice* to keep those packets in the PRISM chipset but the current firmware (mine included) automatically recycles the buffer when the packet is sent. I'm considering *not* recycling the buffer in case the packet needs to be retransmitted and this frees up RAM in the EDTP. (This is one of the reasons I'd like Fred to add EXTRA RAM to the AirDrop.)

So, to answer your question, it depends how you send the data (TCP or UDP) and in which direction and what implementation techniqes you use on the AirDrop.

All in all, complex. And the current EDTP firmware doesn't provide for two packets in flight simultaneously.

This is why TCP is such a complex little beastie, why sockets are liked, why they take up a lot of memory (separate contexts) and why uIP is here (it can do a lot of this in a small amount of RAM).

I'm unsure whether I will pursue the avenue of keeping packets on the PRISM or not. It's a lot of work, probably more than I'm prepared to put in as a hobby project.

Regards,

--Paul

What this signals is *good* news to me, since im running the PIC at 3.3V (~20mhz according to who? i think fred said that) and im doing udp, with application specific ack requiring packets done only intermittently...

my dream is to stream 176 KBytes/sec over this thing...for the above setup do you still see it as being totally out of the question? what if i ran the PIC at higher voltage to get the 40 mhz out of it?

-alex

As stated before I am new to this and trying to learn but I have run into a brick wall with a Very simple project (so I thought). I am using the original drivers for the air-drop-p and the CCS compiler. I need the airdrop to receive serial data on the rs232 port and send it out On the RF side and in reverse I need the data coming from the RF side to come out on the Rs232. All at a very low speed and data rate. This is a project were I receive APRS (Amateur packet radio) data and RF link it to a PDA (telnet) so that I can have all the heavy Equipment in the car and just carry the very portable PDA and see all the aprs data on my PDA-map. (Using win-aprs-ce). If anyone could give me some guidance on this it would be Very much appreciated.

Manfred VA3WK

If you look, Fred has implemented interrupt-driven IO routines using the USART, so you can invoke them.

The tricky thing is that Fred's stack (as is) only allows the PIC to send a packet in response to a packet from the client, so you can only send out your data when the client asks you to. A few ways to get round this:

(1) Make the client poll the server at regular intervals, asking 'have you got data for me' – nasty

(2) Use uIP – not a trivial job, and you'd have to port it to PIC yourself

(3) I am in the process of modding Fred's TCP implementation to allow the server to send packets to the client at its own discretion. I got it working last night and am happy to release it once I have it tidied up a bit (and have incorporated the AirDrop CF drivers)

Julian

— In airdrop_user@yahoogroups.com, "Julian Porter" <j.porter@c...> wrote:
> If you look, Fred has implemented interrupt-driven IO routines
using the USART, so you can invoke them.
Yes i have done that and had no problems with it..

> The tricky thing is that Fred's stack (as is) only allows the PIC to send a packet in response to a packet from the client, so you can only send out your data when the client asks you to. A few ways to get round this:
This is my problem i did not see this (lack of experience i guess)

> (1) Make the client poll the server at regular intervals, asking 'have you got data for me' - nasty
Yes and would not be possible since i can not modify the client.....

> (2) Use uIP - not a trivial job, and you'd have to port it to PIC yourself
Hey may be in a few months(years :-( )

> (3) I am in the process of modding Fred's TCP implementation to allow the server to send packets to the client at its own discretion. I got it working last night and am happy to release it once I have it tidied

up a bit (and have incorporated the AirDrop CF drivers)

Thanks very much Julian again and i think i wait till i see your program. I am also waiting for freds book to come out, there is very little info on this type of tinkering :-)

Manfred

Greetings,

Sounds like a neat project! Keep us posted on your progress please.

Just a question-- is it really required that "telnet" be used over the rf link? One simple thing that comes to mind is UDP. This would be simpler I think.

Quite often, PDAs have support for the "Python" language built in already. And although I am not personally "savy" with Python, I have seen someone write a UDP application in just 10 or 20 lines of Python.

However, I would agree that when it becomes available, the telnet solution that Julian mentioned would be very nice indeed.

William, AB4YD

Programs like CE-Aprs and winaprs have build in capabilities to connect to a server via Telnet. There is no special protocol; data ~~from the TNC is unmodified~~ send to the program That is what made this solution very attractive. Also data send from the program to the TNC is very simple, a data string.... In an emergency you could park the car outside the op-centre and be on the air immediately (no equipment to install). Fast and effective.

Manfred

Seems like an awful lot of negotiation to get a Telnet session up and running just to get a single line of output back; wouldn't something like (shudder) finger be more appropriate? Of course, I guess this is all history, and that's the way it is so you have to live with it...

-- Paul.

what is the current code's ram footprint?
does anyone have a favorite jedec word-addressable sram for interfacing with the PIC?

later
-alex

are there any code sections which either cannot or should not be interrupted (needs interrupts disabled) due to timing issues?

i hope you all take advantage of downtempo radio stations on shoutcast during these late night hacking sessions. it definitely has a positive, mood-stabilizing effect during those times when you've just spent 5 hours on trying to debug your code before realizing you just discovered a bug in the compiler...

hope everyones doing fine
-alex

Here's another idea, gleaned from Linux drivers-- They use only a single BAP. They claim that ping-ponging between the BAPs is not reliable.

Seems worth a try. And it would simplify the code. I can't see where it would hurt performance.
William

> The CIS tells you about how fast you can waggle lines.

But the difference between 3 uS and 12 uS may mean a 4X difference in thruput, right? Seems worth investigating, if someone out there has the equipment and expertise.

William

That's a good idea.. However, in my development cycle I never saw the AirDrop use BAP1..

Glad we are all starting to speak the lingo :-)

Fred

Yeah, whenever I've run through the code it's always using BAP 0.

-- Paul.

Hi everyone.

I am new member here.

I am now seeking the LINUX driver for the PRISM chipset. Can you do me a favor and show me the relevant link to download the driver.

Thanks!

Hi, Everyone

This may sound funny, but I'm considering going to my NDA folks and a lawyer friend asking just how much of the Open Source 802.11b things we all see in the public eye can't I talk about openly... But before I insert foot into mouth, I believe I'll need a bit a ammo.. So, if you can help me that would be nice but I also respect those of you that don't want to get involved. I would like to know of any Linux or otherwise open source documents that you've come across that use the "secret" PRISM terms freely.. I figure if I can get enough scoop that says "everybody on the Internet knows this anyway", maybe I can persuade them to allow us to be considered Open Source as well.. I'm not going to divulge any names of the folks that have something to say.. In fact, you don't even have to post the info if you don't want to. You can just email me directly.

I've personally come across the "document" and some interesting Linux charts telling all about the numbers behind the "document".. HMMMMM...

I don't want to go on a witch hunt, but I would like to have a fair shot at getting the useful truth out about how to use and embed the PRISM chipset with small micros just as the Linux folks have.

Right now this is just a serious thought as I'm really tired of attempting to "hide" the things we all need to make this wireless LAN stuff work.. If you think I need to go back to bed and rest, let me know that too :-)

Fred

Welcome!
Here's the Linux link..

http://www.linux-wlan.org/

Fred

Hi Fred,

Well, the Linux driver is proving very useful. There are lots of magic numbers, of course, but added to that is a bunch of control which makes flow through the driver fairly simple to understand. The next step is to run this on a functioning Linux system. If you replicate the Linux driver functionality, you have a complete system.

-- Paul.

Actually, that is one of several Linux drivers. Personally, I prefer this one:

http://hostap.epitest.fi/

Also there are a couple of Linux drivers that can be confusing, such as the "orinoco" and the "hermes" drivers. While they may support your card, they also support some older cards that are incompatible in subtle ways.

William

As you may know, the "PRISM" chipsets have been around a while. I think the original chipset was from Harris, and you see some part numbers like HFA3841 and HFA3842. They published some pretty detailed documentation freely, with block diagrams and theory of operation paragraphs. You can find them on the web in PDF form, just google for them. They are about 25 pages each. They devote 2 or 3 pages just to the BAP, FID, et al.

Then the chipset went to Intersil, and you find part numbers like ISL3871, ISL3873a, and ISL3874. They also published detailed documentation, freely, which you can find via google. They are 30 to 40 pages each.

Based on how the Linux drivers operate, I would say from the driver writer's point of view, the boards in our Airdrop units are quite compatible with the ones described in the datasheets above, even though our chipset is newer. Indeed, I use my Linux-based IPAQ to make sure that a particular card is compatible. :-)

One area that there doesn't seem to be much information available, other than from the "hostap" Linux driver, is downloading firmware into the CF card from the "host". I personally don't need this capability anyway. But if I did, I would think you could figure it out by careful study of the hostap driver itself.

```
u8_t
uip_eth_poll(void)
{
  unsigned char bnum;
  unsigned i, RxF, len, frame_control, *wp;

  // Debugging feature lets me see the extent of the buffer.
#ifndef NDEBUG
  memset(uip_buf, 0xff, sizeof(uip_buf));
#endif

  // If no packet waiting, return.
  if ((rd_cf_io16(PRISM_EVSTAT_OFF) & PRISM_EVSTAT_RX) == 0)
    return 0;

  // Get the RXFID.
  RxF = rd_cf_io16(PRISM_RXFID_OFF);

  // Get a free buffer access pointer (BAP) to read it.
  // Unsure about this, I reckon I could devote one to reading and
  // one to writing, and even that is pushing the boat out given
  // this software is not interrupt driven.
  bnum = get_free_b();
  switch (bnum)
   {
   case 0:
     // Select RXFID.
     wr_cf_io16(RxF, PRISM_SELECT0_OFF);

     // Prepare to read the frame type from the 802.11 header.
     wr_cf_io16(0x0e, PRISM_OFFSET0_OFF);

     // Wait for offset bit to reset for initial (and only) transfer.
     while (rd_cf_io16(PRISM_OFFSET0_OFF) & PRISM_OFFSET_BUSY)
       ;

     // Read frame type.
     frame_control = rd_cf_io16(PRISM_DATA0_OFF);

     // Write offset to 802.3 data; automatically sets
PRISM_OFFSET_BUSY.
     wr_cf_io16(rxdatalength_offset, PRISM_OFFSET0_OFF);

     // Wait for offset bit to reset for initial transfer.
     while (rd_cf_io16(PRISM_OFFSET0_OFF) & PRISM_OFFSET_BUSY)
```

```
       ;

     // Read the packet length.
     len = rd_cf_io16(PRISM_DATA0_OFF);
     if (len > sizeof(uip_buf)+14)
       len = sizeof(uip_buf)+14;

     // Read the remainder of the packet.
     wp = (unsigned *)&uip_buf;
     for (i = 0; i < (len+14)/2; ++i)  // 2 * mac addresses (12 bytes)
+ length (2 bytes)
       *wp++ = rd_cf_io16(PRISM_DATA0_OFF);

     // Acknowledge the receive event.
     wr_cf_io16(PRISM_EVSTAT_RX, PRISM_EVACK_OFF);

     // If this isn't a RFC1042-encoded frame, don't bother with it.
     if ((frame_control & 0xe000) != 0x2000)
       return 0;

     // Return length of 802.3 packet with SNAP and LLC headers.
     return len+14;
```

That helps tremendously... My next steps are to attempt to get a more detailed datasheet from TRENDnet that may also turn over some leaves... I've got a call into Coexant as well..

Fred

I have a question: do these cards work at 3.3V as well as 5V?

-- Paul.

...ahh, OK, a quick look at the schematic reveals a 3.3V supply.

-- Paul.

According to the TRENDnet datasheet and the SanDisk datasheet 5V is OK too.. I think William had mentioned he had some trouble with a card at 5V not doing what he expected... I haven't tried 5V ops..

Fred

Hi All,

It's strange this one. Here's a function that's in the AirDrop code:

```
void wr_cf_io16(unsigned int data16,unsigned int addr)
{
  char i;
```

Sure thing, not another word other then 802.11x. Which brings me to ask, what uses are there outside of a router and ISP nearby say 100' or less? The coke machine, again, is a good example for a standalone remote microcontroller but it was hooked to a modem to a cell phone to connect to cell tower, then to ISP, etc. I can see the Airdrop devices used around the home, or business that has a router & ISP/LAN, but where else might it be used? Just asking nicely for some examples ...
v/r
Ken Carrigan

```
I'm looking at the AirDrop as a means for remote control using WiFi.
To
that end, there's a project brewing, but it's more diverse than simple
control.  The WiFi adapter is a wireless gateway between things, and
I'm
going to use my AirDrop as a prototype for a device I have in mind.

However, I have another fun project that uses Bluetooth, but that one
is
much further off than the one based on 802.11b.
--
```
Paul Curtis

```
> I would like the charter to restrict this to AirDrop issues, but
there
> are interesting off-topic posts now and again

You guys are right. I regret my previous post.

Please carry on discussing whatever you see fit. I certainly don't
want to squelch discussions here, off-topic (in my view) or otherwise.

I totally agree that the Airdrop, having a CF slot, is great for lots
of things besides/in addition to wifi.

About ethernet-- isn't there a ne2000-compatible driver in the uIP
stack?  I've got such a CF card laying around somewhere. I'll look for
it and let you know the name.
```

William

What this makes me wonder is, is a simple mailing list like this really adequate? I know there aren't many of us (yet), but I can already see strands of activity along these lines:

- wireless comms in general
- embedded 802.11 in general

- Airdrop as embedded CF interface
- Airdrop as 802.11 device
- uIP 802.11
- EDTP TCP/IP stack 802.11
- Discussions of specific architectures (PIC, AVR, possibly MSP430 if I get my act together)

A simple threading mechanism may be a good idea at some point, and should at least help to disrupt any spam we might get in the future. Just a thought.

Julian

PS I hope I don't speak only for myself in finding uIP's API pretty unpleasant. I want a TCP/IP stack which makes as few assumptions as possible about my server-side software, and so uIP is just too complex.

```
Hi Julian,



        From: Julian Porter [mailto:j.porter@...]
        Sent: 08 February 2005 19:50
        To: airdrop_user@yahoogroups.com
        Subject: RE: [airdrop_user] Re: How to convince people about
AirDrop preject


        What this makes me wonder is, is a simple mailing list like
this
really adequate?
```

Lowerpower

--- In airdrop_user@yahoogroups.com, "Paul Curtis" <plc@r...> wrote:
> I don't think there's an NE2000 uIP driver; there may be a custom one
> hanging around, but I don't see a "standard" one. The Socket card is
> based on an NE2000 interface.

I guess we could grab the "driver" from Fred's PacketWhacker project. It is NE2000 compatible.

I don't know if there will be any CF/CIS issues or not. If there are, I'm willing to dig around a bit in some of the Linux drivers to see whats what.

If you don't have a better tool, you could try the little "CIS" dump program I posted a few days ago. That would give us a little to go on. Or if you have Linux, you can use "dump_cis -v" .

William

> I was looking through the CF spec and it seems that the top of the
> card is the part with the lip.
>
> So looking at the pins, the lip is in the back top (not so on the
> wireless cards), the larger of the two slot keys is on the left, and
> pin one is the bottom left pin and pin 26 is the top left pin.
>
> Is this how everyone else reads it? Just checking since most of the
> CF cards I have, have an arrow pointing on the oppisite side which I
> thought might be pin one. Also the lip is usually on the bottom of
> what I'd consider the top of the card (main logo etc.)
>
> Thanx

Greetings,

It is rather confusing for me too. As far as I can tell, there is absolutely no standardization on the little "arrows" and labels on the cards themselves. Perhaps, they are meaningful in their original environment, like maybe the "free" memory card that comes with your digital camera. But my collection of cards seem to have the "arrow" in just about every different direction.

Your description sounds right, except for one thing. The "pins" are

in the socket, the cards themselves have "holes". So, your description is fine with me, if you subsitute the word "hole" everywhere you mention a "pin".

If you happen to have an Airdrop card, pin 1 on the CF connector (mounted on the PCB) is at the corner of the board, and it is the "highest" pin. Pin 26 is directly beneath pin 1, and is closest to the PCB itself.

William

Greetings,

I've added the routines to allocate and read/write memory/buffers on the Prism card. These are the routines that I used for the benchmarking that I mentioned recently.

From what I've been able to tell, the main reason for them being so slow, is the use of the "large memory model".

But even if they are less than optimal, I hope they will clarify how to use the Prism card, and provide some "friendly" names for things.

Until I find a better place, I've put these on my local ISP's site. I hope it work for everyone.

http://tinyurl.com/5öqt

William

The following code successfully associated an Airdrop-A onto an ad-hoc network called SHELBYVILLE running on channel 11. Just for variety, this is written on CrossWorks, but it should be pretty portable. Enjoy.

Julian

```
unsigned char airdrop_config2(unsigned int cmd,unsigned int keyNo,unsigned char *WIFIData)
{
    unsigned int i,j,k;

    switch(cmd)
```

```
            {
                    fr_buffer[k] = WIFIData[i];
                    --j;
            }
            break;
    case WEP_AUTH_SHARED:

            fr_buffer[0]=2;
            fr_buffer[1]=0xfc2a;
            fr_buffer[2]=0;
            break;
    case WEP_AUTH_PRIVATE:

            fr_buffer[0]=2;
            fr_buffer[1]=0xfc2a;
            fr_buffer[2]=1;
            break;
    case WEP_ON:

            fr_buffer[0]=2;
            fr_buffer[1]=0xfc28;
            fr_buffer[2]=1;
            break;,
    case WEP_OFF:

            fr_buffer[0]=2;
            fr_buffer[1]=0xfc28;
            fr_buffer[2]=0;
```

```
            break;
    case WEP_USE_KEY:

            fr_buffer[0]=2;
            fr_buffer[1]=0xfc23;
            fr_buffer[2]=keyNo;
            break;
    case WEP_SET_KEY_64:

            fr_buffer[0]=4;
            fr_buffer[1]=0xfc24+keyNo;
            fr_buffer[2]=(WIFIData[1]<<8)|(WIFIData[0]);
            fr_buffer[3]=(WIFIData[3]<<8)|(WIFIData[2]);
            fr_buffer[4]=WIFIData[4];



            break;
    case WEP_SET_KEY_128:

            fr_buffer[0]=8;
            fr_buffer[1]=0xfc24;
            fr_buffer[2]=(WIFIData[1]<<8)|WIFIData[0];
            fr_buffer[3]=(WIFIData[3]<<8)|WIFIData[2];
            fr_buffer[4]=(WIFIData[5]<<8)|WIFIData[4];
            fr_buffer[5]=(WIFIData[7]<<8)|WIFIData[6];
            fr_buffer[6]=(WIFIData[9]<<8)|WIFIData[8];
            fr_buffer[7]=(WIFIData[11]<<8)|WIFIData[10];
            fr_buffer[8]=WIFIData[12];
```

so 1.25million instructions per second at 128000, 64000, or 32000 bytes_sent gives me either 39, 78, or 156 _instructions_ per each byte i need to send!

is this anywhere close to what i need?

thanks.
-Alex

Greetings Alex,

Sounds like an interesting project. All I can say is, it isn't impossible, but you sure would be close to the limit, on your highest data rate.

Any chance you could use compression? It seems overkill to have 32bits per sample at 32khz sample rate.

The Airdrop's interface to the Prism card is 8 bits, but you must always send an even number of bytes.

Dont forget that even UDP has a header of its own (you could turn off the checksum of the UDP data), then you have the IP header, and the ethernet header. Fortunately, very few of those fields would be changing for every packet, so they could be re-used (and not re-written every time). I'd make the packets as large as possible, so that the need to deal with headers doesn't happen so often.

Yes, you can just send the bytes over as slowly as you like, as long as you send two bytes at a time.

I think I like your idea of polling for the A/D sample and storing it directly in the Prism memory.

But what happens when you need to stop and deal with the UDP, IP and ethernet headers? And issue the Transmit command, etc? It could be tricky without interrupts. But not impossible.

I assume your application is simplex (one-way).

William

bvwelch <bvwelch@hotmail.com> wrote:

> it isn't impossible, but you sure would be close to the limit, on
> your highest data rate.

Did you mean my lowest? My highest was 128Kbytes/second transmission, with only 39 instructions in between each byte's arrival in the SPI buffer to grab that byte and send it to the cf card? My lowest was 32kbytes/second with 156 instructions in between each byte's arrival.

Incidentally, after thinking about it, it seems more reasonable to do 44.1kbytes per second and send 22,050Hz mono. thats my aim as of right now. which by the previous math gives me 113 instructions per byte. is 113 instructions enough?

> Any chance you could use compression? It seems overkill to have
> 32bits per sample at 32khz sample rate.

data compression would be nice, but there is no processing power for that without, well, adding another processor or changing processors altogether. Thats not likely in the immediate future but is a possibility on revision 2 of the device.

As for the 32bits...that comes from the fact that its stereo (2 16 bit channels). if i decide to do mono then i only have 16bits per sample.

> The Airdrop's interface to the Prism card is 8 bits, but you must
> always send an even number of bytes.

sending an even number of bytes is cool for me because the way the stero a/d works is that it send me 4 bytes in each sample period, 2 for left channel and 2 for right channel. they spi buffer fills and sets a flag on every byte, so i get two bytes (1 channel) then have 2 bytes worth of "liesure time"...

but then how many total instructions does it take to send 2 bytes back-to-back?

> Dont forget that even UDP has a header of its own (you could turn
> off the checksum of the UDP data), then you have the IP header,
> and the ethernet header.

regarding the UDP header...definitely you are right that most stuff would never change, in fact as far as i can tell only the checksum would change, and i can either choose to ignore it, or i can update the sum as each sample comes in, and the last 2 bytes i would send before issuing a transmit would be the checksum.

> make the packets as large as possible, so that the need to deal
> with headers doesn't happen so often.

yep id probably want 512 bytes of data per packet...whats the limit i can store in card, incidentally?

> But what happens when you need to stop and deal with the UDP, IP
> and ethernet headers? And issue the Transmit command, etc? It
> could be tricky without interrupts. But not impossible.

I suspect, but I haven't proven it yet, that the Prism card is so fast, as compared to our picmicro, that we only need 1 buffer. But at most, we could have 2 buffers and "ping-pong" between them-- One belongs to us, and we are filling it, while the other belongs to the Prism card, and is being transmitted.

William

Hi everyone,

I'm not sure if anyone is interested, nor how much use this will be but....

I have access to a bunch of Ethernet cards that are going to scrap.

No biggie there, but the MAC addresses are all "official" Intel or 3com addresses.

I was wondering if anyone is interested in any of the numbers?.

If so, I could pass along the MAC addresses on a one by one basis to whoever wants them and make sure the card actually does get trashed.

It does mean that you'd be able to use the Airdrop on the net without "bumping into" another MAC address (all those "00EDTP" addresses are going to bump into each other sooner or later)

Is this of interest to anyone, or am I missing something (again)

Let me know

Michael

Greetings,

I'd be interested in a few of those addresses, and if any of the boards are of the old 8-bit ISA, "ne2000 compatible" variety, there might be some interest in a few of those also. Some friends and I built a few little embedded computers with ethernet, and we need mac addresses for them. Mine is based on the EDTP "packet whacker" which is ne2000 compatible. But some folks might want to actually hack the ISA card itself to use for a project.

About the Airdrop, I believe we obtain the MAC address from within the Prism card itself, so we should be OK with a legit MAC address. My wifi card has it written on a little sticker on the back of the card, and it also prints out in the CIS, and also when you ask for the Mac address via the Prism commands.

Thanks,

William

The MAC address on an Airdrop is that embedded within the WLAN NIC, so it already is official.

However, there could certainly be value in having some for those of us using EDTP wired products (e.g. Easy-Ethernet, Packet Whacker, etc). I'd certainly be interested in acquiring some (4 or 5 – I have a lot of embedded Ethernet devices!)

Julian

The MAC address on an Airdrop is that embedded within the WLAN NIC, so it already is official.

However, there could certainly be value in having some for those of us using EDTP wired products (e.g. Easy-Ethernet, Packet Whacker, etc). I'd certainly be interested in acquiring some (4 or 5 – I have a lot of embedded Ethernet devices!)

Julian

```
> >  I don't believe that this is possible.  My reading is
> that once you
> > issue TRANSMIT you cannot touch the buffer until the transmit is
> > complete and you receive acknowledgement (or auto-recycle the FID).
>
>
> Actually, if I understand the Airdrop code, it initializes 3
> buffers at the beginning, and there are some fields in the
> buffer that get re-used over and over without being re-written.
>
> So, I don't there would be a problem with assuming/depending
> on the contents of the buffers.
```

>
> I won't know for certain until I test this theory out though.

The buffers certainly can be "reused" but you can't start writing to them before you're told the firmware has completed its time with them. The function "get_free_TxF" is used for this purpose--if the PRISM firmware isn't done with one of the buffers, you can't touch it, and get_free_TxF will return you another buffer to use. I agree some of the
fields that are used are static and the PRISM firmware probably won't update them, but I'm pretty sure you won't be able to deposit a second packet into RAM into the same buffer that's actively being used by the PRISM for transmission.

--
Paul

Why did I call you Brian? No idea. Sorry.

--
Paul

The buffers certainly can be "reused" but you can't start writing to
> them before you're told the firmware has completed its time with
them.

Right, agreed. So having 2 buffers and ping-ponging between them should work. But I suspect, but haven't proven it yet, that the prism card is so fast, that he "completes" his usage of the buffer before I can really get around to checking for it. Just a hunch. And probably not safe/robust to depend on it.

William

how does long is the usual turnaround time between issuing a transmission and receiving an acknowledgement?

cheers
-alex

Written for CROSSWORKS again. Works pretty well. The PRISM spec is not entirely accurate, so I had to fudge things a bit.

Julian

```
{
 unsigned int nRecords,i,j;
 unsigned int *p;
 r_read(0xfd88);

 if(fr_buffer[0]<4) return;
 nRecords=(fr_buffer[0]-3)/31;
 printf("Scan returns %i results, with reason %i
(%i)\n",nRecords,fr_buffer[3],fr_buffer[0]);
 for(i=0;i<nRecords;i++)
 {
  p=&fr_buffer[4+31*i];
  printf("Ch %2i Noise %3i SNR %3i ",p[0],p[1],p[2]);
  printf("BSSID %04x%04x%04x Bcn int %5i Cap %04x ",p[3],p[4],p[5],p[6],p[7]);
  printf("SSID(%2i) ",p[8]);
  for(j=0;j<16;j++)
  {
   if(2*j<p[8]) printf("%c",0xff&(p[9+j]));
   else printf("_");
   if(2*j+1<p[8]) printf("%c",0xff&(p[9+j]>>8));
   else printf("_");
  }
  printf(" Rates %04x %04x %04x %04x %04x SRT
%04x\n",p[25],p[26],p[27],p[28],p[29],p[30]);
 }

 fr_buffer[0]=3;
```

please help me out. Thanks.

Sincerely,
S.Ranganathan

Question (1) re IPs is answered in my first reply to you – the IP enters into the code only when packets are recognised at the IP stack layer, so yes it can vary dynamically.

Question (2): obviously you can have a wired NIC and a wireless one on the same box, but why do you want to? It makes things much more complex to try to run two NICs simultaneously. In your scenario, why not have the PC a wireless device itself, so it either talks to the master, which then talks to the slaves, or it does an IP multicast (for which you'll need to tweak some of the PRISM settings in Fred's code)?

[illegible]

Hi all...
I've been trying to figure out how to increase the frame size for the ~~data getting~~ put out on the wire. It seems no matter what I do, I can't get it to spit out more than 60 bytes. I know I must be missing something obvious. If someone can point me in the right direction I would appreciate it. Thanks.

Jeff

Greetings,

I've written some routines to read and write "RIDs" with "friendly" names. There are routines to display the various firmware revision info (inside the Prism card). Also a routine connect to the BSS (access point) and display its Mac address, and monitor the link status.

Until I find a better place, I've put these on my local ISP's site. I hope it will work for everyone.

http://tinyurl.com/58ojt

William

did you already post something to give the link quality (Signal to Noise ratio)?
I'd love to have something like that.

-alex

Yes I did, on 3 February.

Hi Julian,

Have you looked into the received frames from the AP to see what rate

each of them were transmitted at? For the life of me, I keep decoding the frames I receive as transmitted at 1Mbps which I think isn't right, but don't know whether it is or not. The AirDrop is only a few meters away from the AP, so I would expect a higher bps.

Regards,

-- Paul.

Paul,

I haven't done that yet. However, when I was playing with ad Hoc I never managed to get a connection at more than 5.5 Mbps, and the airdrop was only a couple of metres from the card.

Julian

Hmm, ok. I assume there's some XP drivers for the TRENDnet card? If I whack it into a CF to PCMCIA converter, do you reckon I can get it work work on the PC? That way I can see how fast it thinks the connection is? I've no data to substantiate the 1mbps rate, I suppose I can have a
go at trying out a round-trip time using PING, but with only a single packet outstanding using uIP, it's going to be slow. I'll need to do some more work to qualitatively measure the link speed.

Rgds,
Paul

Paul,

I'll have a look in the Prism spec and see what it says about rates. The rate information that is extracted in the scan routine I posted should tell you something about the connection, but I couldn't (immediately) find anything to tell me how to interpret it.

I assume the TRENDnet card (as it's PRISM) will just have the standard Windows driver -- of course you won't have any config apps. If you have a Linksys WC12, at least I know it comes with a disk with XP config code (though I've never used it).

I mean rebuild pcb for PCMCIA connector.
There is no connector such as, but you can find some extender card
for that purpose like
this "http://www.teampctechnology.com/product_detail.php?id=39#"

Calvin

You need to do more than change a couple of defines--for instance the
LLH size changes as does a couple of other things (such as the format
of
some of the structures).

I've already done the uIP port to the AirDrop and it'll be making an
appearance on our web pages when I get back from Dallas.

Regards,

--
Paul

--- In airdrop_user@yahoogroups.com, "Paul Curtis" <plc@r...> wrote:
> I've already done the uIP port to the AirDrop and it'll be making an
> appearance on our web pages when I get back from Dallas.

I noticed a reference to the uIP supporting the RTL8019. That chip is
NE2000 compatible, and is the same one used on the EDTP
PacketWhacker.

So maybe uIP will work with your CF ethernet card already.

William

> where do you buy PCMCIA-> CF connector? I have seen CF->PCMCIA but
> not the other way arround!

This is the one I have. Please note, I have not had a chance to test
it out yet.

http://www.semsons.com/comflastopcc.html

William

...yep, it's certainly something to try. I dropped by Ed's pages
yesterday and noticed the compatibility statement.

Rgds,

-- Paul.

...at the sale price, I reckon it's a bargain and doesn't matter if it
doesn't work...

I'll order one. :-)

--
Paul Curtis, Rowley Associates Ltd http://www.rowley.co.uk
CrossWorks for MSP430, ARM, AVR and (soon) MAXQ processors