ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΑΣΥΡΜΑΤΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΣΕ ΚΑΡΤΕΣ ΑΠΟΜΑΚΡΥΣΜΕΝΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΟΜΒΩΝ ΠΕΙΡΑΜΑΤΙΚΩΝ ΔΙΑΤΑΞΕΩΝ

Διπλωματική διατριβή της

Σταματίας Αφροδίτης Ηλιούδη

Επιβλέπων: Λέανδρος Τασιούλας, Καθηγητής Π.Θ.

Συνεπιβλέπων: Αθανάσιος Κοράκη, Λέκτορας Π.Θ.

Volos, September 2013

Η έγκριση της διπλωματικής διατριβής από το Τμήμα των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Θεσσαλίας δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

(Ν. 5343/32, Άρθρο 202, παρ. 2)

Ευχαριστίες

Η παρούσα διατριβή εκπονήθηκε στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστήμιου Θεσσαλίας.

Ολοκληρώνοντας την διπλωματική διατριβή μου θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή του τμήματος ΗΜΜΥ της Πολυτεχνικής Σχολής του Π.Θ., Τασιούλα Λέανδρο, καθώς και τον συνεπιβλέποντα λέκτορα του τμήματος ΗΜΜΥ της Πολυτεχνικής Σχολής του Π.Θ., Κοράκη Αθανάσιο, για την πολύτιμη καθοδήγηση τους, την ενθάρρυνση και την συμβολή τους στην εκπόνηση και ολοκλήρωση της διπλωματικής διατριβής στην περιοχή των ασυρμάτων δικτύων υπολογιστικών συστημάτων.

Επίσης θα ήθελα να ευχαριστήσω θερμά τον υποψήφιο διδάκτορα του τμήματος ΗΜΜΥ της Πολυτεχνικής Σχολής του Π.Θ., Καζδαρίδη Ιωάννη, για την βοήθειά τους κατά την διάρκεια των εργαστηριακών δοκιμών.

Επιπρόσθετα θα ήθελα να ευχαριστήσω τον κ. Ηγούμενο Ιωάννη, για την στήριξή του και την πολύτιμη συνεισφορά του στην εκτέλεση των πειραμάτων.

Θα ήθελα να ευχαριστήσω όσους με συμπαραστάθηκαν και με βοήθησαν με οποιοδήποτε τρόπο στην διάρκεια των σπουδών μου και της διατριβής μου.

Τέλος θέλω να εκφράσω την ευγνωμοσύνη μου στους γονείς μου για την συνεχή στήριξη, ενθάρρυνση και αμέριστη αγάπη τους κατά την διάρκεια των σπουδών μου.

Πρόλογος

Σε γενικές γραμμές οι κόμβοι του δικτύου, συμπεριλαμβανομένων των σταθμών και συσκευών δικτύου δύσκολα μπορούν να ρυθμιστούν σωστά. Για να ελαχιστοποιηθεί η απαιτούμενη διαμόρφωση ή τα σφάλματα ασφαλείας, οι διαχειριστές δικτύων συνήθως υλοποιούν ένα δίκτυο υπολογιστών μικρότερης κλίμακας (δίκτυο για εργαστηριακές δοκιμές) εξομοιώνοντας το πραγματικό δίκτυο, όπου και δοκιμάζονται οι απαιτούμενες αλλαγές των ρυθμίσεων. Ωστόσο, για να λειτουργεί ένα δίκτυο υπολογιστών χρειάζεται συχνά τοπική διαχείριση και υποστήριξη, που μερικές φορές είναι δαπανηρή και ανέφικτη διαδικασία. Η ικανότητα της απομακρυσμένης διαχείρισης είναι ένα σημαντικό χαρακτηριστικό πλεονέκτημα, που επηρεάζει τη συνολική απόδοση ενός δικτύου υπολογιστών.

Οι τεχνολογίες απομακρυσμένου ελέγχου μπορεί να επιτρέψουν σε ένα διαχειριστή συστήματος να συνδεθεί απευθείας σε κόμβους του δικτύου, ιδιαίτερα διακομιστές (servers) και εξοπλισμό δικτύωσης με σκοπό την υποστήριξη ή αλλαγή της διαμόρφωσης, όπου χρησιμοποιείται σπάνια ο περίπλοκος εξοπλισμός των δοκιμών. Δεδομένου ότι ο εξοπλισμός του δικτύου είναι γεωγραφικά κατανεμημένος, η πρόσβαση των σημαντικών τμημάτων του δικτύου επιτρέπει την εξ αποστάσεως πιο αποτελεσματική χρήση του εξοπλισμού του δικτύου, οπουδήποτε, μειώνοντας το κόστος κατασκευής και λειτουργίας του δικτύου. Όπως στην περίπτωση ενός σύννεφου διακομιστών (server cloud), ένας διαχειριστής μπορεί να ζητήσει πρόσβαση σε συσκευές δικτύου (routers και switches) απομακρυσμένα και να συνδεθεί με αυτές μέσω γραφικού περιβάλλοντος χρήστη (Graphical User Interface-GUI) ή μέσω υπηρεσιών διαδικτύου (web services). Σήμερα έχουν αναπτυχθεί διάφορες εφαρμογές και υπηρεσίες απομακρυσμένου ελέγχου, που κατά την άποψη των κατασκευαστών διαθέτουν διαφορετικές και μοναδικές δυνατότητες. Πέρα από την εξοικονόμηση κόστους, ορισμένες από αυτές τις εφαρμογές επιφέρουν πολλά πρόσθετα οφέλη, συμπεριλαμβανομένης της δυνατότητας να αυτοματοποιούν πλήρως την διαμόρφωση του δικτύου και τις απαιτούμενες δοκιμές.

Ο σκοπός της παρούσας εργασίας είναι να σχεδιαστεί και να δοκιμαστεί μια κατασκευή βασισμένη σε μικροελεγκτή, που θα μπορούσε να επιτρέπει την απομακρυσμένη διαχείριση των κόμβων μίας δοκιμαστικής πλατφόρμας ασύρματου δικτύου. Η πρωτότυπη κάρτα βασίζεται στον μικροελεγκτή ATmega328 της εταιρείας Atmel εξοπλισμένη με ένα M10 GSM Modem της εταιρείας Quectel και δοκιμάστηκε ειδικά για το NITOS (Network Implementation Testbed using Open Source platforms) Wireless Testbed, το οποίο αναπτύχθηκε στο εξωτερικό του κτιρίου Γκλαβάνη του Πανεπιστημίου Θεσσαλίας (Π.Θ.). Επίσης η εφαρμογή διαχείρισης δικτύων εξ αποστάσεως δοκιμάστηκε εναλλακτικά χρησιμοποιώντας ένα Wiznet Ethernet ελεγκτή. Επιπρόσθετα, υλοποιήθηκε μια απλή σειριακή επικοινωνία χρησιμοποιώντας την ίδια κάρτα, που βασίζεται στον μικροελεγκτή ATmega328 της Atmel, μέσω ενός κατάλληλου ολοκληρωμένου κυκλώματος MAX232 (MAX232 IC). Όλα τα πειράματα επικοινωνίας πραγματοποιήθηκαν στο εργαστήριο NIT (Network Implementation Testbed Lab-NIT Lab) του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών στο Πανεπιστήμιο Θεσσαλίας (Βόλος, Ελλάδα). Επιπλέον, η παρούσα εργασία αναλύει τις τρέχουσες τάσεις της βιομηχανία σχετικά με τον απομακρυσμένο έλεγχο και προσδιορίζει τα χαρακτηριστικά, που όχι μόνο θα επιτρέπουν στους διαχειριστές να ελέγχουν τους κόμβους του δικτύου, αλλά να τους επιτρέπεται επίσης να διαχειριστούν το συνολικό δίκτυο από μια κεντρική θέση.

Λέξεις Κλειδιά – Ασύρματα Δίκτυα (Wireless Networks - WN); NITOS Testbed Platform; Απομακρυσμένος Έλεγχος Δικτύου (Remote Network Control); Global System for Mobile communication (GSM); Απομακρυσμένη Διαχείριση Δικτύου Testbed (Remote Network Testbed Management-RNTM); Σειριακή Επικοινωνία (Serial Communication); Σύνδεση RS232 (RS232 Interface)

UNIVERSITY OF THESSALY
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΑΣΥΡΜΑΤΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΣΕ ΚΑΡΤΕΣ ΑΠΟΜΑΚΡΥΣΜΕΝΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΟΜΒΩΝ ΠΕΙΡΑΜΑΤΙΚΩΝ ΔΙΑΤΑΞΕΩΝ

Dissertation

Σταματίας Αφροδίτης Ηλιούδη

Supervisor: Leandros Tassioulas, Prof. of UTH

Co-Supervisor: Athanasios Korakis, Lecturer of UTH

Volos, September 2013

Acknowledgments

This Dissertation was prepared at the Department of Electrical and Computer Engineering, Faculty of Engineering of the University of Thessaly.

Completing the diploma thesis I would like to thank the supervisor of the ECE Department, Faculty of Engineering, University of Thessaly, Professor Dr Leandros Tassioulas, and the co-supervisors Lecturer of the ECE Depurtment, Faculty of Engineering, University of Thessaly, Lecturer Dr Korakis Athanasios, for their valuable guidance, encouragement and their contribution to the development and completion of the thesis in the area of wireless networks.

I would also like to thank the doctoral student of ECE Department, Faculty of Engineering, University of Thessaly, Kazdaridis Ioannis, for his help during the laboratory tests.

Additionally I would like to thank Mr Igoumenos Ioannis, for his support and valuable contribution in carrying out experiments.

I would like to thank those who stood by with me and helped me in any way during my studies.

Finally I want to express my gratitude to my parents for their continuous support, encouragement and unconditional love during my studies and preparation of my thesis.

Abstract

In general network nodes including stations and network devices are difficult to configure correctly. To minimize configuration or security errors, network administrators usually build a smaller scale computer network (test lab) simulating the real network and test out the required configuration changes. However operating a computer network needs frequently local management and support, which is expensive and sometimes impractical. The ability of remote management is an important feature affecting the overall performance of a computer network.

Remote control technologies can enable a system administrator to connect directly to network nodes, especially servers and net equipment for support or configuration changes using rarely complicated test equipment. Since the network equipment is geographically distributed, access of important network parts remotely allows more efficient use of network equipment anywhere reducing the building and operating costs of the network. Similar to a server cloud, an administrator could request access to networking devices (routers or switches) remotely and connect them through GUI or web services interface. Nowadays there are developed several remote control applications and services available that all claim to have different, unique capabilities. Beyond saving costs, some of them bring about many additional benefits, including the ability to fully automate network configuration and testing.

The aim of the present thesis is to design and test a microcontroller based construction that could allow remote node management of a testbed. Prototype card is based on an Atmel ATmega328 microcontroller equipped with a Quectel M10 Modem and it was specifically tested for NITOS Wireless Testbed deployed at the exterior of the University of Thessaly (UTH) campus building. Remote network management application is also using a Wiznet Ethernet. Additionally a simple serial communication is also implemented using the same card based on Atmel ATmega328 microcontroller though a MAX232 IC. All the communication experiments are implemented in NIT lab of the Department of Electrical and Computer Engineering at University of Thessaly (UTH, Volos, Greece). Additionally presented work analyses current trends in the remote control industry and identifies features that will not only allow users to control network nodes, but can also enable them to manage an entire network from a centralized location.

Keywords - Wireless Networks (WN); NITOS Testbed Platform; Remote Network Control; Remote Network Testbed Management (RNTM); Global System for Mobile communication (GSM); Serial Communication; RS232 Interface).

Contents

1 Introduction

1.1 Background of the study

Networking improves communication within a specific area moving large amounts of information from place to place in a reliable way. Computer networks allow more efficient work to be accomplished in a specific time. However it is reported that most network outages are caused by operator errors in configuration, rather than networking equipment failures [6], [14]. Sometimes it is necessary for the network administrators even if they are informed soon, to get to a specific location in order to solve the related problem. Most of the times, this anxiety could be avoided if they at least have access to the Internet to be able to change node configuration or to do something for fixing the occurred failure. A lot of proposed solutions are based on remote network control management. Remote control software is one way in which an organization or a company can expand its network usability and centrally administer the overall network. This links to any network node required being controlled. Administrators can keep surveillance over a computer network to ensure its operating conditions, while having the power to immediately access being informed about the failure taken place in a network node and fix occurred problems related to network [8]. This helps keep the network operating at maximum potential as networks could be technically supported in a matter of sort time without a requirement of moving up to problematic locations. Among the approaches based on remote network control management some of them are using Short Message Service (SMS) by means of telecommunications networks [10]. In these implementations the administrators could be informed for network operating conditions or failures and allowed to access the network sending commands for altering operating conditions or fixing the occurred problems. This is achieved through an application, which is set up on one or more of the manageable network systems using SNMP (Simple Network Management Protocol) protocol. The needed information for the network administrators is gathered via Internet, while network managers can send commands to the application software running on a system by means of short messages (SMS) to let them fix the failure.

Using a remote management scheme, the network inspection (concerning performance and security issues), configuration and repair are getting more time efficient. The right remote network control approach makes all this possible, while creating a barricade of security against network attacks. A remote control session is conducted, when two necessary modules are present:

a. The Administrator Machine Application (Controller)
b. The Controlled Machine Application (Network Node), which grants the controller access to the network node.

Normally remote management should be able to do every task except some specific tasks, which must be operated only from node location. The network node simply allows the controller to control or access it.

Extensive researches on remote network management systems in laboratories as well as at industrial sites have been conducted and various control methods have been developed so far. Within the large volume of such researches, a lot practical solutions have been given to the remote control problem

applied on a variety network structures improving the communication performance and data transmission efficiency.

This dissertation address issues of remote control techniques for networking with special attention to solutions based on easy implementations using simple commercial hardware and software schemes.

1.2 Objectives of the study

This research is expected to contribute a new practical technique to the design and implementation of a wireless based remote network tested management for NITOS (Network Implementation Testbed using Open Source code) Wireless Testbed, which is deployed at the department of Electrical and Computer Engineering in University of Thessaly (UTH). On this purpose, an experimental prototype system was designed, built and tested in the NIT Lab at University of Thessaly (UTH). In the tested system, both hardware and software were used in combination to verify the proposed remote network management procedure.

The main objectives of present work include:

- Develop a remote network control scheme that is easily implemented in a low-cost based on commercial microcontroller equipped with a GSM Modem and an Ethernet network controller as additional communication option.
- Investigate developed remote management scheme in terms of network efficiency and performance operating at real time.
- Develop an alternative solution for remote network management using standard serial communication ports interfacing microcontroller TTL signals with RS232 standards in the case of network node on site.

1.3 Dissertation Organization

The dissertation is organized as follows:

Chapter 1 introduces the background for present dissertation research, the significance of the study and the research objectives.

Chapter 2 reviews the state of the art and recent developments of remote network management. Also, the developed hardware scheme is presented briefly considering the communication platform of NITOS Wireless Testbed.

Chapter 3 addresses the development of required hardware for network node management. Moreover, basic operation principles of components used such as Arduino microcontroller board and Wiznet Ethernet interface are discussed whereupon remote network control is briefly explored.

Chapter 4 presents an alternative scheme for remote network management based on serial communication. The main aspects discussed for the developed scheme are the interfacing of microcontroller with RS232 standard port and the conversion of TTL to RS232 signals with respect to MAX232 IC.

Chapter 5 proposes a simple implementation of remote network management using simple commercial hardware devices. The communication is conducted by means of browser-based remote sessions over HTTP or HTTPS ports.

Chapter 6 gives the overall conclusions of the work and recommendations for future work.

2 Remote Management for NITOS Wireless testbed

2.1 Types of Remote Control Management

Considering remote control sessions, these could be mainly conducted in two primary ways:

a. Browser-Based Remote Sessions
b. Software-Based Remote Sessions

Function of Browser-Based Remote Sessions is implemented through an Internet browser that generally requires the temporary download of network node software (client software) necessary for a connection to be made. Software-Based Remote Sessions operate using individual control and node or client applications installed separately on each machine. Both ways offer advantages and disadvantages that will ultimately determine which type of remote control solution is best for specific networking environment [11].

2.1.1 Browser–Based Remote Management

Browser-based remote control sessions are conducted through a website that utilizes an ActiveX version of a control application with a downloadable node or client application that is usually required for the remote control session to take place. These browser based remote control solutions typically “piggy-back” over HTTP or HTTPS (TCP ports 80 and 443 respectively), and must connect to a third party website to establish a connection. A web server based application can be configured to require authentication before allowing access to the page, where the ActiveX control component is loaded and roles for users can be created from within the management console. With ActiveX Remote Sessions, the majority of the security will come from the client being able to disconnect the control whenever they’d like and the flexibility is almost unparalleled. Because the control application is pre-configured on the server side, there will be no control configuration or installation necessary by the administrator, a huge advantage if administrators move around and connect from any box they happen to be working with or if clients are typically “on the road” and need support over the Internet. This type of remote control allows the control piece to connect directly to the desired client from anywhere on the Internet [11].

2.1.2 Software-Based Remote Management

Quality software-based remote control solutions provide enterprises with more features, security and configuration options than web-based or ActiveX solutions for a private network environment. With software-based remote control, both control and client applications are present and the control can securely connect directly to the desktop of the client machine within the private network.

Software-based remote control solutions differ from browser-based or ActiveX solutions because the client and control applications are installed on both machines allowing a remote session to take place. A good software-based remote control will enable an administrator to deploy, install, and configure controls and clients from a centralized location. Such a feature saves valuable time of IT administrators and decreases the overall cost as this can be done in a matter of minutes.

The ability to remote control distant computers is no new technology. Since its development almost three decades ago, several products have evolved beyond simple remote control capabilities to offer

feature-rich applications that simplify helpdesk environments. Providing with a number of extremely powerful utilities, a good remote control application will enable system administrators to take complete control of their entire network without having to leave their working place [11].

2.2 Testbed platforms

Experimentally driven research is the key to success in exploring the possible futures of the Internet. The OneLab initiative provides an open, general-purpose, shared experimental facility, both large-scale and sustainable, which allows European industry and academia to innovate and assess the performance of their solutions. Based on the results of several different European and national projects, OneLab offers access to a range of tools and testbeds including PlanetLab Europe, the NITOS wireless testbed, and other federated testbeds [12].

The Network Implementation Testbed Laboratory (NITLab) of the Computer and Communication Engineering Department at University of Thessaly is also affiliated with the Center for Research & Technology Hellas (CERTH). CERTH, as part of the OneLab project, collaborates with other European institutes and it is in the process of federating NITOS with other testbed facilities, in particular PlanetLab Europe, providing in this way access to a unified European experimental infrastructure. The research of the lab focuses on the design, study and implementation of wireless schemes and their performance in the real environment. In this context, NITLab has developed a testbed named NITOS, which stands for Network Implementation Testbed using Open Source code [1], [2].

2.2.1 NITOS Wireless Testbed - Network Implementation Testbed Laboratory (NIT Lab)

As wireless networks rapidly gain significance in the world of Information and Communication Technologies (ICT), it is essential to test and evaluate tomorrow's wireless protocols and technologies in real-life experimental facilities. NITOS (Network Implementation Testbed using Open Source code) is a wireless experimental testbed that is designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real-world settings. It has been developed in the city of Volos, Greece by OneLab partner CERTH, in association with NITLab, the Network Implementation Testbed Laboratory of the Computer and Communication Engineering Department at the University of Thessaly. CERTH (Center for Research and Technology Hellas) has developed a wireless testbed called Network Implementation Testbed using Open Source platforms (NITOS). NITOS is a testbed offered by NITLab and consists of wireless nodes based on open source software. The testbed is designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real world settings. NITOS wireless testbed has been developed as a part of the wireless facilities of the European project OneLab2 [1], [2].

NITOS consists of nodes based on commercial Wi-Fi cards and Linux-based open-source platforms, which are deployed both inside and outside of the University of Thessaly's campus building, as illustrated in Fig. 1 below. NITOS testbed currently consists of 50 operational wireless nodes, which are based on commercial Wifi cards and Linux open source drivers. The testbed is designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real world settings. Currently, three kinds of nodes are supported: orbit-like nodes (shown in yellow), diskless Alix2c2 PCEngines nodes (shown in blue), and GNU/MIMO nodes (shown in green). The control

and management of the tesbed is done using the cOntrol and Management Framework (OMF) open-source software. NITOS testbed is deployed at the exterior of the University of Thessaly (UTH) campus building (see Fig. 1). In testbed configuration there are two servers with discrete operations to perform. The first one, the webserver, is responsible for scheduling of the testbed's resources and its primary task is to handle users' requests to access the testbed. It also keeps valuable information for experimentation and developing through a wiki-site. The second one, the console-server, is used to run experiments on the testbed through OMF, and to host various network services including DHCP, DNS, NTP, TFTP, PXE, Frisbee, NFS, MySQL, OML and Apache. The webserver communicates with console and they share common info and data for experiment and user handling. Indeed the webserver is the interface between the testbed and the external world, and the console server subsequently is used to execute experiments in the testbed. The console server is connected through a PoE switch which enables power switching to all nodes. Currently, the testbed supports orbit-like nodes (figure 2), diskless nodes (figure 3), and some custom made nodes. In particular, there are 10 orbit-like nodes, 15 diskless ones (ALIX 2c2) and 20 custom made already deployed in the campus testbed. Besides this development, NITOS has a branch for experimenting with GNU-radios, and uses 6 USRPs equipped with XCVR2450 daughterboards. This choice was driven by the fact that communication norms should be as close to IEEE802.11 standards while experimenting with Software Defined Radio.

NITOS is remotely accessible and gives users the opportunity to implement their protocols and study their behaviour in a real-case environment. The NITOS platform is open to any researchers who would like to test their protocols in a real-life wireless network. They are given the opportunity to implement their protocols and study their behavior in a custom tailor-made environment. NITLab is constantly in the process of extending its Testbed capabilities. Users can perform their experiments by reserving slices (nodes, frequency spectrum) of the testbed through NITOS scheduler that together with OMF management framework, support ease of use for experimentation and code development. OMF simplifies the procedure of experiment defining and offers a more centralized way of deploying experiments and retrieving measurements. Detailed information about NITOS testbed use can be found on the NITOS Site. Instructions for using the NITOS scheduler are also available online.

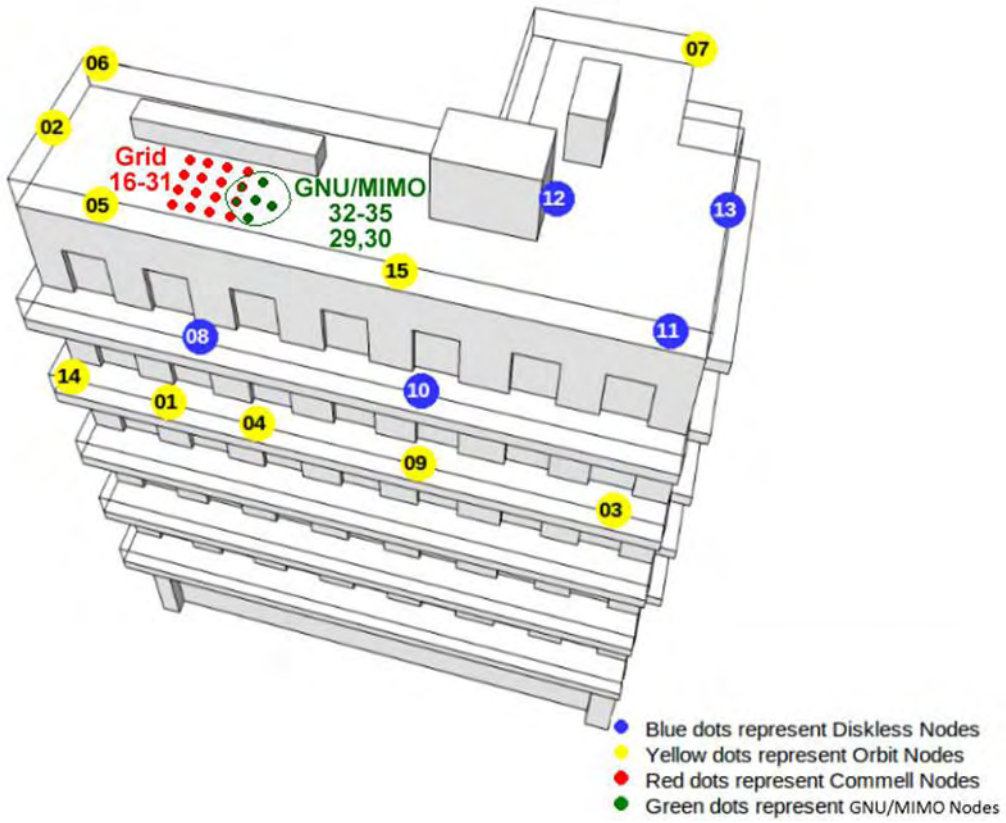


Figure 1 Exterior of Glavani Building's testbed deployment at the UTH [1]

Through NITOS scheduler a more sophisticated way of node allocation to users is achieved since multiple users have the opportunity to run their experiments at the same time sharing the testbed's resources. In addition, users are prevented from interfering with each other by selecting different frequency spectrum to operate. NITOS scheduler is an innovative testbed framework that supports a more effective way of scheduling. NITOS Scheduler is a tool responsible for managing the testbed resources. NITOS nodes are laying in a six-floor building, while it is planned to extend it to other buildings. Therefore, NITOS topology would not be realistically represented on a two-dimension shape; this is why the scheduler is letting the user to see the exact position of each node in the building.

2.2.2 NITOS Testbed Deployment

In Fig. 1, it is illustrated the exterior view of Glavani Building where the NITOS testbed is deployed. NITOS testbed already supports 10 Orbit nodes, 5 diskless nodes and 20 Commell nodes. Analytical details about the specifications of the nodes are explained below, in Fig. 2.

Figure 2 NITOS Testbed of Glavani Building's testbed deployment at the University of Thessaly[1]

The nodes are powered over Ethernet; however, for the orbit-like nodes that are regular powered through 220 Voltage, using a custom made electro-mechanical disruptor controlled via the aforementioned switch and its configurable interface. All type of nodes are equipped with wireless interfaces for experimenting through the air and particularly use Wistron CM9 - mPCI Atheros 802.11a/b/g 2.4 & 5 GHz cards that also take advantage of the extensions of MadWifi driver so that Click modular router features can be exploited.

The wired infrastructure of NITOS testbed deployment is used for control experimentation and resource management via OMF (cOntrol and Management Framework). OMF was initially developed by ORBIT at Rutgers University and is now maintained and extended by NICTA and CERTH. NITOS, as a stand-alone testbed, is constantly in the process of extending its testbed capabilities. In the long term, NITOS intends to expand its deployment with new nodes and new technologies (WIMAX, 3G) and develop new control and management modules for efficient experimentation on wireless networks. Additionally, NITOS, as part of Onelab 2, collaborates with other European institutes and intends to provide access to a unified experimental infrastructure, considering federation among different testbeds. To this end, a primary goal is to federate with the PlanetLab Europe testbed.

CERTH has developed a wireless testbed called Network Implementation Testbed using Open Source platforms (NITOS). NITOS is a testbed offered by NITLab and consists of wireless nodes based on open source software. NITOS wireless testbed has been developed as a part of the wireless facilities of the European project OneLab2. NITOS registered users can run their experiments real-time on the configurable wireless nodes. For quick information about using the testbed, you can refer to tutorial. If you already are an experienced user, you can go to scheduler to begin the node booking process. Furthermore, the scheduler mirrors the testbed's slicing capabilities. Until now, spectrum slicing support is enabled on NITOS. This means that various users may use the testbed at the same time, without

interfering with each other, since each one of them will be using different spectrum. Each user at scheduling declares this spectrum. The scheduler does not allow for a user to choose any channel that has been chosen by another user. Although NITOS is supporting spectrum slicing, the testbed reservation procedure has been simplified to the user that visits the site for the first time [1], [2].

2.3 Application Using Browser-Based Remote Control Sessions

One of the main aims of the thesis is to design a card that will have remote management of the nodes of NITOS testbed illustrated in Fig. 1. In present work the remote network management is achieved applying Browser-Based Remote Sessions method via a wireless GSM module. Remote control sessions are conducted through an Internet browser sending HTTP Requests, while the controlled node is running an HTTP Server application configured properly for management requirements replying HTTP Responses. The monitoring of the node is been implemented using the RS 232 interface. For this network management implementation the following devices are used:

a. Arduino Ethernet - a microcontroller board based on the Atmel ATmega328 microcontroller including Wiznet Ethernet interface [6]
b. Arduino GSM shield, including radio modem M10 by Quectel
c. MAX232 - an IC interfacing microcontroller TTL signals to RS232 standards for serial communication purposes [3]

2.4 Summary

Two major aspects on the computer network issues have been reviewed comprehensively in terms of remote network management and testbed platforms (NITOS Wireless Testbed). The related types of remote management are briefly discussed and provided to outline the state of the art and potential trends of technology development in this area. Also NITOS Wireless Testbed fundamentals and basic operation ideas are introduced. Moreover an application using software-based remote control sessions is briefly discussed in this chapter for conducting a server node of the NITOS Testbed network

3 Prototype card

3.1 Arduino Platform

For the impelentation of this remote management solution we use an Arduino board as master of the interfaces which are implemented on different modules. In this chapter we describe each of these modules.

3.1.1 AVR Architecture

The Arduino platform used in the experimental work is based on Atmel ATmega328 microprocessor. This is an 8-bit microprocessor built according to Harvard architecture. However, it provides general purpose registers of 16-bit, although it is equipped with 8-bit data bus [13]. A detailed microprocessor diagram is illustrated in Fig. 3

The ATmega328 microcontroller is an upgrade from the very popular ATmega8. They are pin compatible, but not functionally compatible. The ATmega328 has 32kB of flash, where the ATmega8 has 8kB. Other differences are in the timers, additional SRAM and EEPROM, the addition of pin change interrupts, and a divide by 8 prescaler for the system clock.

The schematic below shows the Atmel ATmega328 circuit (see Fig. 4) as it was built on the test board. The power supply is common and is shared between all of the microcontrollers on the board. The ATmega328 is in a minimal circuit. It is using its internal 8 MHz RC oscillator (divided by 8). With the ATmega328 it is needed to both burn a bootloader and download Arduino sketches. The bootloader is programmed using the ISP programming connector, and the Arduino sketches are uploaded via the 6-pin header. Programming the Arduino bootloader into the ATmega88, ATmega168, or ATmega328 micrcontroller will change the clock fuses, requiring the addition of an external crystal. The crystal shown on the schematic is only required when the ATmega328 is going to be used as an Arduino, although it may be desired in any real world application (see Fig. 4). Typically ATmega328 is running at 16 MHz, but it will run as high as 20 MHz.

There are schematics and an ExpressPCB design file for a full 28-pin AVR development board that supports the ATmega8, ATmega48, ATmega88, ATmega168, and ATmega328 (the ATmega328 pinout is the same as these others)

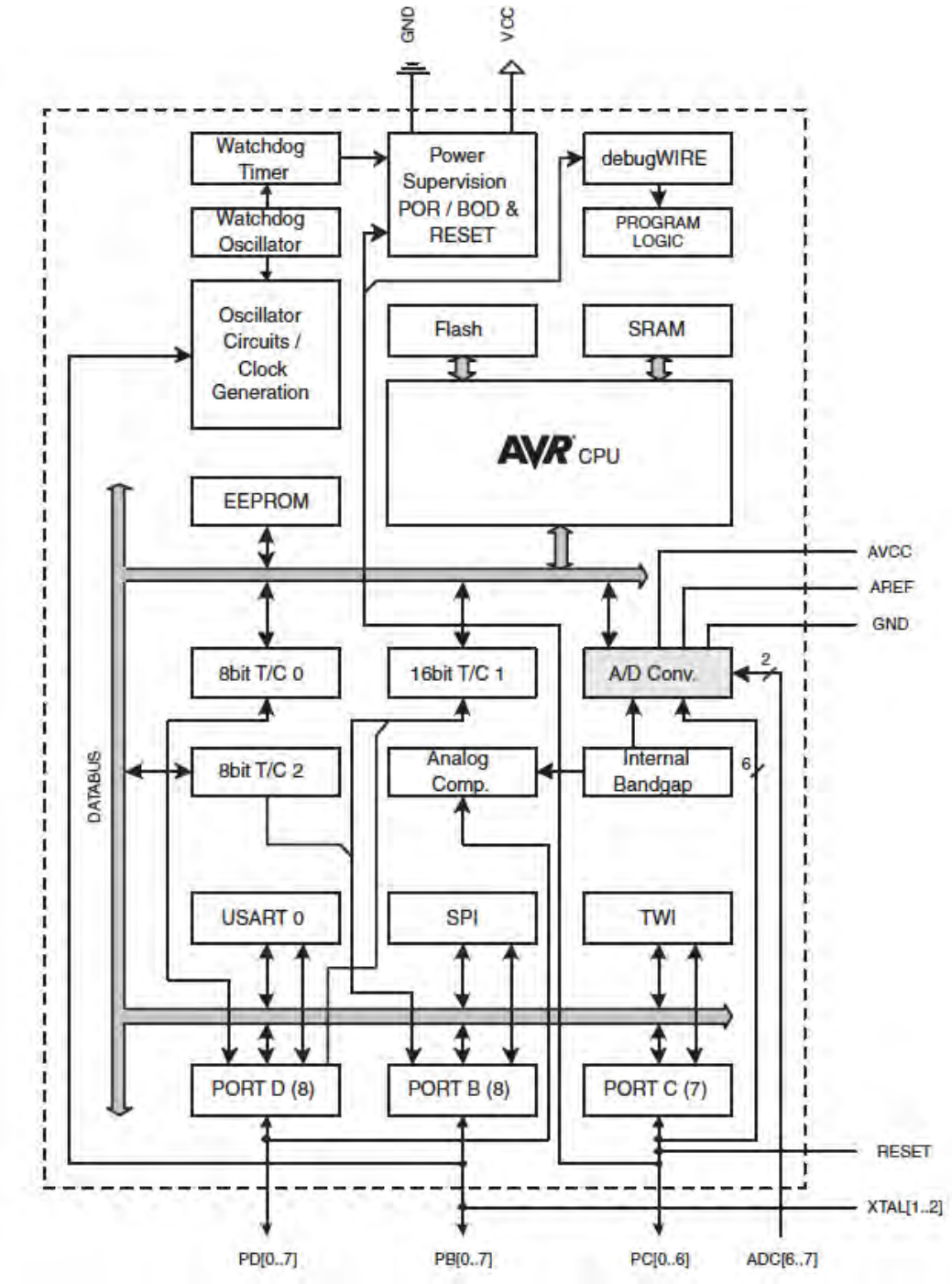


Figure 3 Architecture of Atmel ATmega328 in details [13]

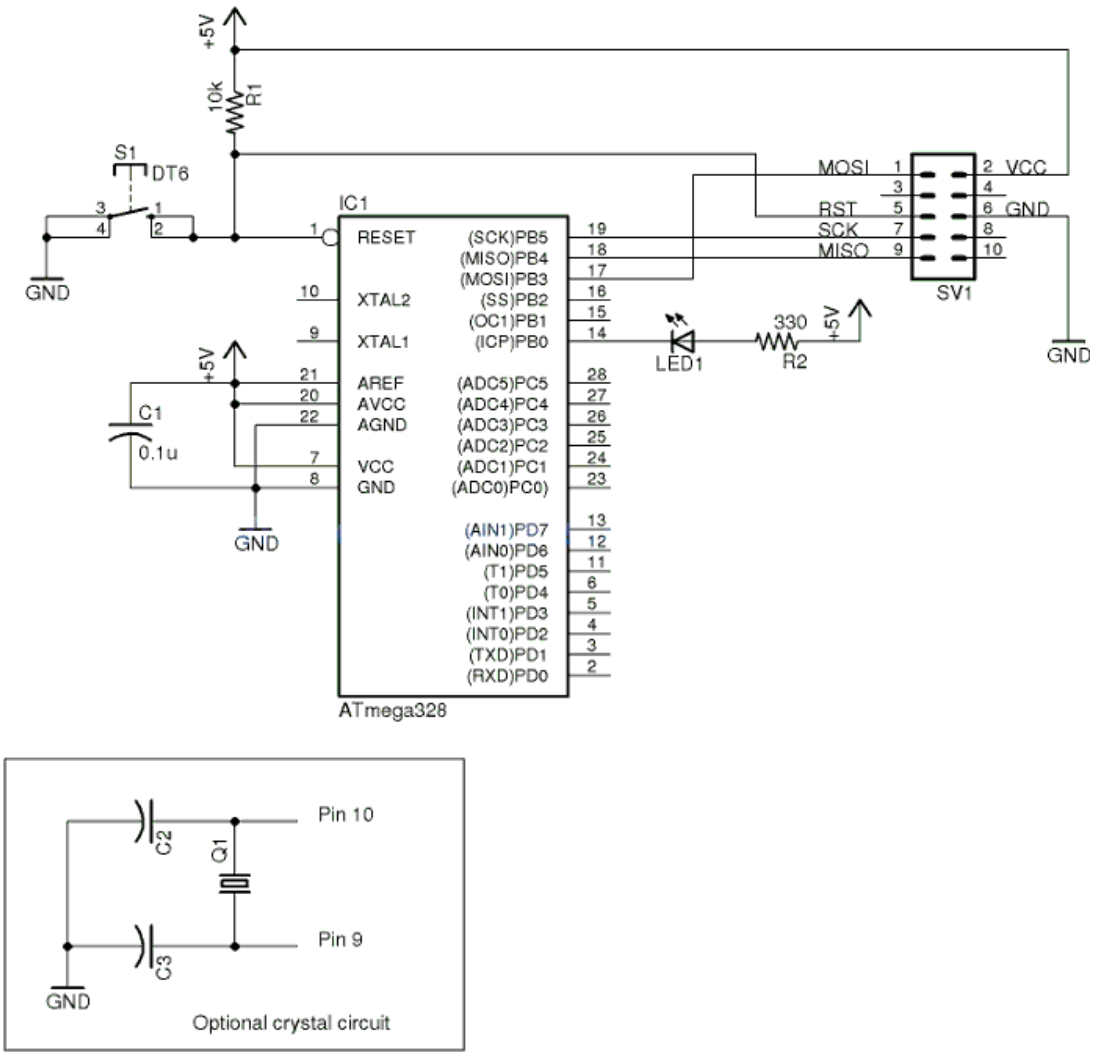


Figure 4 Atmel ATmega328 schematic diagram [14]

3.1.2 GSM Module

GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile) is a standard set developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobile phones. It became the de facto global standard for mobile communications with over 80% market share.

The GSM standard was developed as a replacement for first generation (1G) analog cellular networks, and originally described a digital, circuit switched network optimised for full duplex voice telephony. This was expanded over time to include data communications, first by circuit switched transport, then packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution or EGPRS) [22].

GSM standard module allows a PC or other device to control a GSM phone via standard and extended AT commands. In present case an Arduino board is equipped with GSM for remote network

management providing wireless communication. In the case of an Arduino GSM shield those commands are been transmitted through pins 2 and 3.

Figure 5 An Arduino GSM shield with a Quectel modem mounted

3.1.3 Ethernet Interface

An alternative solution for remote network management is implemented using ATmega328 board equipped with a Wiznet Ethernet interface. The Arduino Ethernet Shield allows an Arduino board to connect to the Internet. It is based on the Wiznet W5100 ethernet chip (datasheet). The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Ethernet library must be used to write sketches, which connect, to the Internet using the shield. The ethernet shield connects to an Arduino board using long wire-wrap headers, which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top. Communication is done through SPI (Serial Peripheral Interface). For this reason, the board is reserved the pins 10-13. SPI also communicates with SD (Secure Digital) controller, with the SS pin 4.

3.1.3.1 Description of Serial Peripheral Interface (SPI) Signals

SPI Bus is a synchronous serial data link that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame [16]. More analytically the SPI bus specifies four logic signals:

SCLK: Serial Clock (output from master);

MOSI: Master Output, Slave Input (output from master);

MISO: Master Input, Slave Output (output from slave);

SS: Slave Select (active low, output from master).

As shown in Fig. 6, each device has one I/O interface and, in case of a slave operation, SPI can ignore the input signals as long as the master does not select it.

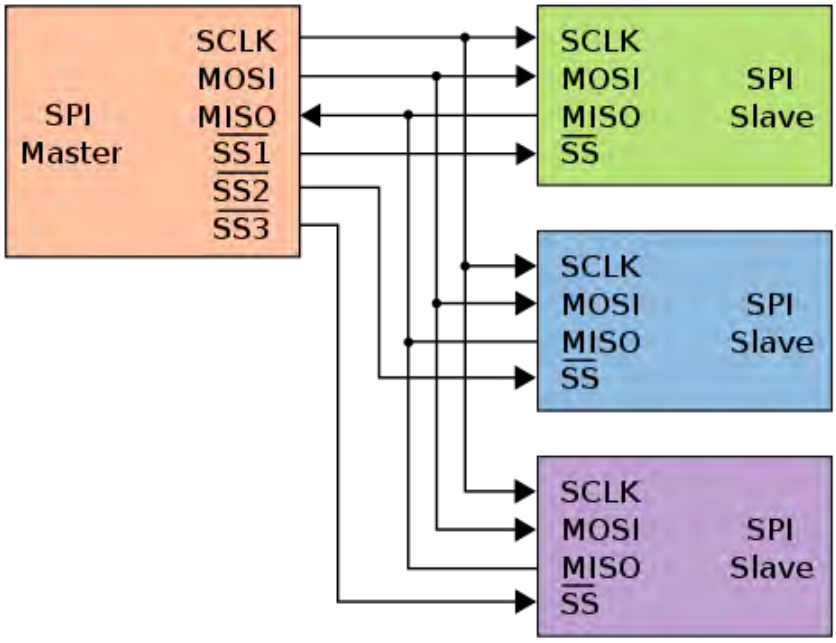


Figure 6 Typical SPI bus configuration with three slave devices connected

3.1.3.2 Pros/Cons

Advantages:

- Full duplex communication
- Higher throughput than I²C or SMBus
- Complete protocol flexibility for the bits transferred
  - Not limited to 8-bit words
  - Arbitrary choice of message size, content, and purpose
- Extremely simple hardware interfacing
  - Typically lower power requirements than I²C or SMBus due to less circuitry (including pull up resistors)
  - No arbitration or associated failure modes
  - Slaves use the master's clock, and don't need precision oscillators
  - Slaves don't need a unique address — unlike I²C or GPIB or SCSI
  - Transceivers are not needed
- Uses only four pins on IC packages, and wires in board layouts or connectors, much fewer than parallel interfaces
- At most one unique bus signal per device (chip select); all others are shared
- Signals are unidirectional allowing for easy Galvanic isolation
- Not limited to any maximum clock speed, enabling potentially high throughput

Disadvantages:

- Requires more pins on IC packages than I²C, even in the three-wire variant
- No in-band addressing; out-of-band chip select signals are required on shared buses
- No hardware flow control by the slave (but the master can delay the next clock edge to slow the transfer rate)
- No hardware slave acknowledgment (the master could be transmitting to nowhere and not knowing it)
- Supports only one master device
- No error-checking protocol is defined
- Generally prone to noise spikes causing faulty communication
- Without a formal standard, validating conformance is not possible
- Only handles short distances compared to RS-232, RS-485, or CAN-bus
- Many existing variations, making it difficult to find development tools like host adapters that support those variations
- SPI does not support hot plugging (dynamically adding nodes)

3.1.4 Serial Interface

For interfacing RS232 with TTL logic (5V for logic 1, 0V for logic 0) of Arduino board in a simple way an integrated circuit MAX232 IC is used. MAX232 is one of the many IC in the market, which implements conversion between RS232 -/+10V and TTL +/- 5V. In brief, it is a simple voltage level converter. Its simple charge pump design allows the circuit to generate +/-10V from a 5V supply using four capacitors. This charge pump can double up the supply voltage for RS232 transmitter eliminating the need to design a power supply for +/-10V. This chip converts RS232 signal voltage levels to TTL voltage levels and vice-versa enabling Arduino board to communicate to a PC through its serial ports (COM1 or COM2).

3.2 Prototype Shield Overview

The overall control system consists of an Arduino Prototyping Shield, a GSM Modem card, a Wiznet card and a MAX232 IC. In the experimental setup a breadboard is built for the MAX232 IC interfacing. The usage of a breadboard for testing is a good idea, since fixing mistakes on soldered prototype board can be frustrating and messy. Breadboarding allows testing the communications and designing the final prototype shield. It is known that this way of testing is great for sketching out an electronics idea, but it is terrible at storing a more permanent prototype. Implemented breadboarding for the MAX232 IC interfacing is demonstrated in Fig. 7. Creating permanent prototypes for use with Arduino or other microcontroller development boards should include a share of the Arduino pinouts. In general Prototyping Shield makes easy the design custom circuits.

An alternative design could include the Prototype Shield of RS232 Interface shown in Fig. 8. Prototype Shield of RS232 Interface could be used in remote control management scheme after experiments completed in breadboard. Arduino Prototyping Shield has got extra connections for all of the Arduino I/O pins and space to mount through-hole and surface mount integrated circuits such as MAX2323 IC. This allows making custom circuits and Arduino into a single module in a convenient way.

Wiznet card is an Ethernet shield and it can be stacked as any other shield) on top of the Arduino Prototyping Shield, since this kit provides stackable header pins. Some other development boards, such

as Netduino, require 3.3V analog signals instead of the Arduino's 5V. The signal voltage could be selected between either 5V or 3.3V by means of a jumper provided.

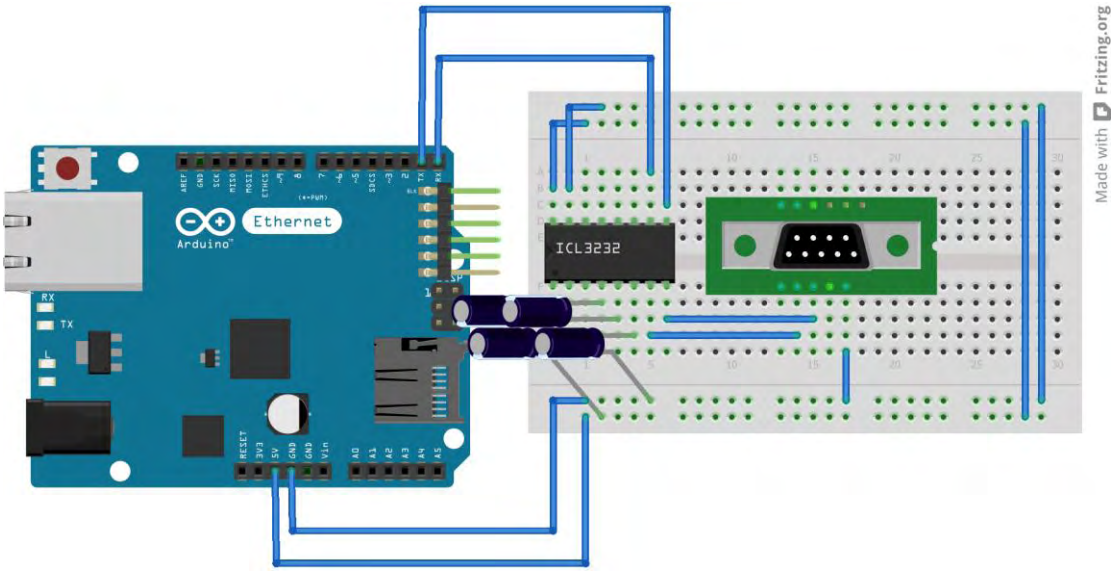


Figure 7 Breadboard for MAX232 IC connecting to Ethernet Board

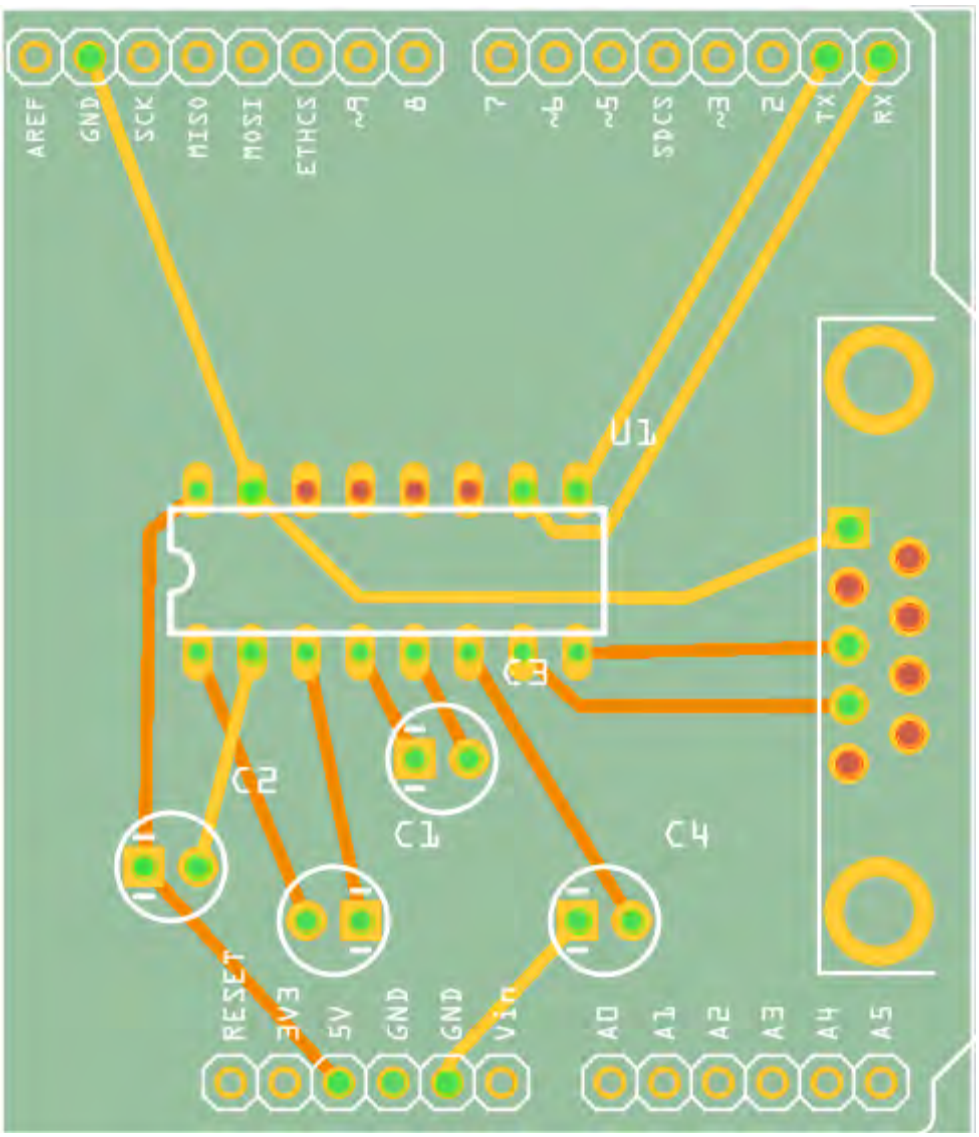


Figure 8 Prototype Shield of RS232 Interface used in Remote Control Management experiments

3.3 Summary

In this chapter, the Arduino platform used in the experimental work is discussed. Here a GSM module is connected with Arduino board to implement the wireless communication for remote management. A detailed diagram of Atmel ATmega328 microprocessor is also presented and its circuitry is reviewed. Two additional remote management schemes are discussed by means of Ethernet (Wiznet Ethernet) or Serial (RS232) interface. Mainly communication based on Ethernet is accomplished by means of SPI (Serial Peripheral Interface). Also advantages and disadvantages of SPI communication modes are discussed focusing on data transfer efficiency. Additionally serial communication is presented by means of a program example written in C++ programming language. Finally Prototype Shield and a Schematic are described analytically showing the overall system connections in details.

4 Implementation of remote management

4.1 Programming the Arduino board

Arduino board is programmed using the Arduino Integrated Development Environment (IDE), which is a cross-platform application written in Java. It is similar to the IDE for the Processing programming language and the Wiring projects. Arduino programs are written in C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need to define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings.
- loop(): a function called repeatedly until the board powers off or resets.

When the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple main() function at the bottom, to make it a valid C++ program. The Arduino IDE uses the GNU tool chain and AVR Libc to compile programs, and uses avrdude command lines to upload the created hex file into the Target AVR.

Arduino Programs are basically C++. Mainly its program structure consists of the following tree parts:

a. Header: declarations, includes, etc
b. b. setup()
c. c. loop()

The setup() function is like Verilog's "initial" and it is executed once when program starts. Finally the third part called loop() is like Verilog's "always". C++ code is continuously re-executed when the end of loop() is reached until we have a reset signal.

4.2 Reviewing the Source Code

4.2.1 Arduino Libraries included

Four main libraries are included at the beginning of the final program, all written in C++:

SPI.h is used for the connection with the Ethernet and SD peripheral, based on SPI protocol.

Ethernet.h includes the implementation of many functions of classes used such as Ethernet and Client. It also implements the function that initiates the serial connection (SPI) between the microcontroller and the Ethernet peripheral, using SPI.h.

GSM.h is used for the GSM module interface. The GSM shield receives AT commands from the dedicated serial. GSM defines functions that implement these AT commands in a more human-friendly form.

SD.h defines all the functions needed to allow the microcontroller talk with the sd controller over SPI

In order to include all the above functions the space of RAM needed reaches approximately 36MB.

4.2.2 Global variables

- MAC address of the network interface,
- IP address pre-set, in order to know which node operates,
- Instance of a EthernetServer listening to a dedicated port
- Instance of a GSM interface and SMS stream
- Instance of a stream to a file we use to write in the SD card

4.2.3 Functions

The following functionalities are implemented inside setup():

- initializing of the Serial session and waiting for establishment,
- initializing of the Ethernet interface,
- initializing of the HTTP server,
- initializing of the GSM server,
- a “ready” message to the serial bus,

The following functions are called continuously by loop():

http_serv() which checks every time if there is an http client available. If there is

- prints a Serial message
- while the HTTP session is active it fetches one by one chars from the client buffer and searches for the first “/” (the argument of HTTP GET follows), and calls decode() with client as parameter
- when the decode() returns the loop breaks (no multiple instructions functionality) and closes the HTTP session

gsm_serv() which checks if there is any sms available by the modem. If there is it calls decode() with the sms stream as parameter

decode() which implements a lexical analyser. It takes as parameter a stream and it parses it based on the Finite State Machine (FSM) at Fig.9

The following functions are called by decode(), depending on the state:

print_page() which, based on the message it receives as argument, creates the proper http post. The message can be either a file or a string.

screen_printer() which collects the input from the serial interface and stores it into a file.

4.3 Description of the Web Server

In order to call Ethernet.begin() we have to pass as parameter the MAC address of our Ethernet Interface and a prefered IP - since this Web Server has remote control purpose and an ip address set by DHCP will not be as useful as a pre-set. Begin() is called in the setup() function. In the loop() function we check for available client and, if so, we read the HTTP Request character-by-character by client [15].

A detailed listing of entire program is presented in APPENDIX A.

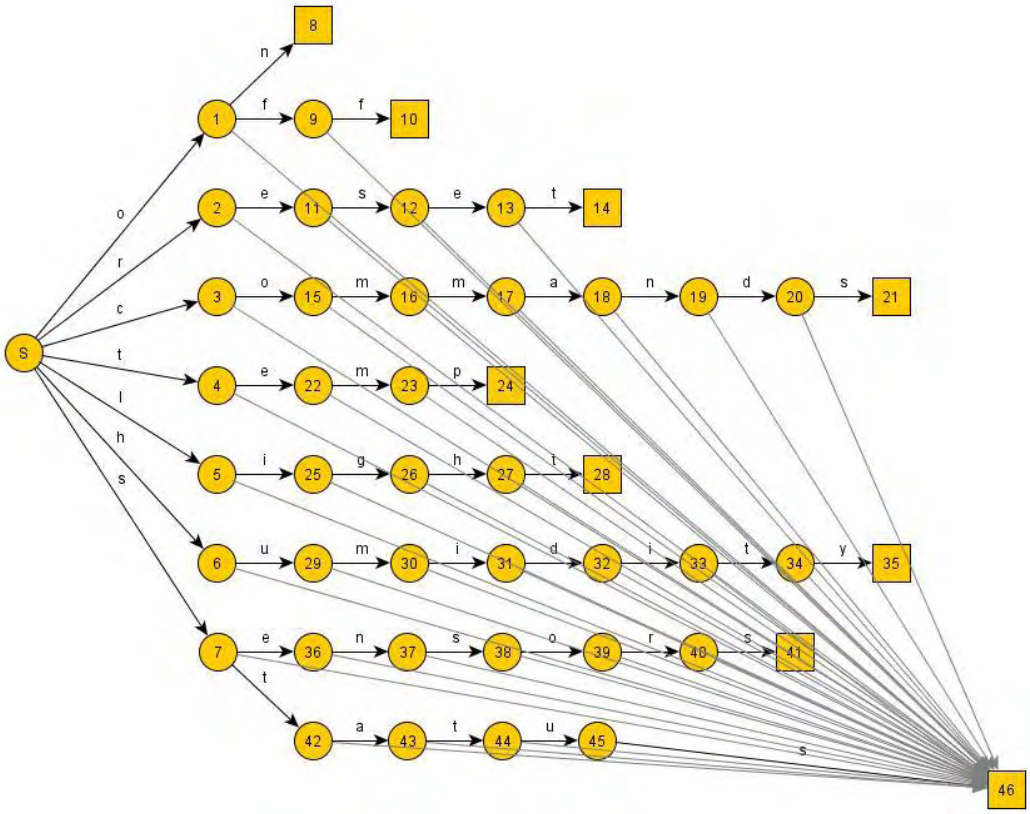


Figure 9 A detailed graphical representation of Lexical Analyser with interconnections

4.4 Summary

In this chapter a description of the developed web server program is analyzed using C++ programming language. Communications between ATmega328 microcontroller board and a network node (e.g. a server) is also illustrated through the analytical comments presented. The tests of the prototype card are presented in the next sections.

5 Experimental setup and testing

5.1 Practical work

The experimental setup consists of an RS232 interface (MAX232), an Ethernet card based on Wiznet Ethernet controller (10/100Mbps) with an SD card inside the SD slot and a GSM modem for wireless communication (Quectel M10). These connection interfaces are already described in previous sections (see Chapter 3). Remote computer network management is achieved through the developed software for any of the two ways, via Ethernet or wirelessly via GSM modem. Enabling parameters included in the executed program determine which hardware interface (serial RS232, Ethernet or Wireless GSM) is active for communication defining the operation mode.

5.2 Experimental Remote Management in Steps

Executing the program written in C++, there are two options to manage the remote network nodes in the following steps:

1. **Input:** HTTP requests to the IP address <assigned id> requesting one of the commands of the command set (<assigned id>/on)
   Output: Command in capitals displayed on the monitor
2. **Input:** SMS with body one of the commands in the command set
   Output: Command in capitals displayed on the monitor

An extra feature is to have a prospective of the screen of the terminal. The output of the screen is collected in a file stored in the SD card. The content of the file can be seen remotely in the following steps:

1. **Input**: HTTP requests to the IP address <assigned id> requesting sc (<assigned id>/sc)
2. **Output**: HTTP response displaying the output of the screen.

5.3 Summary

In this chapter a discussion on experiments completed using the hardware equipment and running the developed web server program. Remote communication could be obtained through any hardware interface, i.e. serially, directly via Ethernet or wirelessly via GSM modem. Experiments show that the developed web server software operates quickly and exceptionally well allowing access to a remote node. Therefore this hardware-software implementation could be applied to manage remotely network nodes in a fast and easy manner. The prototype card has been tested running the remote management program for web server.

6 Conclusion and Future Work

6.1 Conclusions

This dissertation covers major issues and solutions dealing with remote computer network management over Internet. Related remote node control methods have been studied and the following conclusions can be presented upon the fulfilment of the dissertation:

a. A remote network control scheme is designed, implemented and tested using a low-cost commercial microcontroller board equipped with a GSM modem. For management purposes a relatively small C++ program is also developed in a communication environment using HTTP protocol. Since the network the GSM communication uses does not allow SMS transmittion without balance, this remote control interface is limited to only sending commands.
b. Additionally solution for remote network management has been proposed using standard Ethernet network controller and serial communication port. These interfaces have been tested and could be applied as alternative concepts of remote connection to extend the communication capabilities of the proposed management system.
c. Other contributions in this dissertation include:
    1 optimisation of the running program code written in C++
    2 minimisation of the code execution time without decreasing its comprehensive skills.

Compared to conventional remote control solutions, the proposed one features the flexibility to design parameters of microprocessor based communications with low cost hardware equipment and easy C++ programming.

6.2 Future Work

Future research may be of interest concentrating on the following:

a. Include an automated procedure into running program to test remotely network nodes and estimate network performance.
b. Explore the possibility of extending the proposed remote management design adding more peripherals for diagnosis purposes.
c. Implement this remote control using an Arduino Mega and taking advantage of the bigger RAM memory.

References

[1] http://nitlab.inf.uth.gr/NITlab/index.php/testbed

[2] http://nitlab.inf.uth.gr/NITlab/index.php/testbed/hardware/testbed/testbed-deployment

[3] avrfreaks.net and C Programming for Microcontrollers

[4] Atmel Data Sheets, http://www.atmel.com/Images/doc2543.pdf

[5] http://arduino.cc/en/Reference/Ethernet

[6] Arduino Ethernet, http://arduino.cc/en/Main/ArduinoBoardEthernet

[7] http://en.wikipedia.org/wiki/Lexical_analysis

[8] Erik Nordstrom, Per Gunningberg and Henrik Lundgren, "A Testbed and Methodology for Experimental Evaluation of Wireless Mobile Ad hoc Networks", *In Proceedings of the 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom 2005),* pp. 100-109, Trento, Italy, February 23-25, 2005.

[9] Huan Liu and Dan Orban, "Remote Network Labs: An On-Demand Network Cloud for Configuration Testing", *Proceedings of the WREN'09,* pp. 93-101, August 21, 2009, Barcelona, Spain.

[10] Ghasemzadeh, M. and Foroushani, V.A., "Remote management of computer networks by short message service", Proceedings of International Conference on Computer and Communication Engineering 2008, ICCCE 2008, pp. 300 – 305, Kuala Lumpur, 13-15 May2008.

[11] CrossTec Corporation, "Beyond Remote Control-Features that Take Remote Control Capabilities to the Next Level of Network Management", 500 NE Spanish River Blvd, FL, USA, http://www.crossteccorp.com/remotecontrol/.

[12] OneLab FUTURE INTERNET TESTLABS, http://www.onelab.eu/

[13] Washigton umiversity lecture notes notes; Lecture 6 - Introduction to Atmega328 and Arduino, Arduino C++ language; Lecture 7 – Controlling Time, CSE P567, http://www.cs.washington.edu/education/courses/csep567/10wi/lectures/Lecture6.pdf.

[14] http://avrprogrammers.com/atmega328bd.php

[15] http://en.wikipedia.org/wiki/Lexical_analysis

[16] http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[17] http://arduino.cc/en/Reference/Ethernet

[18] http://arduino.cc/en/Main/ArduinoBoardEthernet).

[19] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, "Compilers: Principles, Techniques, and Tools (commonly known as the *Dragon Book*)" , 2nd edition, PEARSON Addison Wesley, Pearson Education, Boston, USA, 2007.

[20] http://en.wikipedia.org/wiki/Lexical_analysis.

[21] http://www.dreamincode.net/forums/topic/260592-an-introduction-to-compiler-design-part-i-lexical-analysis/.

[22] http://en.wikipedia.org/wiki/GSM

Appendix A – Source Code

A detailed source code listing is presented in the following pages. Useful comments are embedded in to the source code for better explanation of program operation. The entire C++ code is divided into five main parts described as follows:

```
#include <SPI.h>

#include <Ethernet.h>

#include <SD.h>

#include <GSM.h>


#define GSM_ENABLE


/* DHCP was not supported by Ethernet.h since ver.1.0

** In our case we need pre-set IPs in order to know which node we operate */

byte mac[] = { 0x90, 0x2A, 0xDA, 0x0D, 0x39, 0x11 };

IPAddress ip(10,64,45,164);


//By default http uses port 80

EthernetServer server(80);


#ifdef GSM_ENABLE

GSM gsmAccess;

GSM_SMS sms;

#endif


#ifdef SD_ENABLE

#define SPACE 0x20

File fscreen;

#endif
```

```
/* This part runs the first time the board powers on
or after RESET signal */
void setup() {

  // start serial interface for the RS232
  Serial.begin(115200);

  // wait until serial is established
  while (!Serial) {
  }

  //start the Ethernet interface
  Ethernet.begin(mac, ip);
  //Ethernet.begin(mac);

#ifdef GSM_ENABLE
  // start GSM shield
  gsmAccess.begin();
  sms.flush();
#endif

#ifdef SD_ENABLE
  pinMode(10, OUTPUT);
  if (!SD.begin(4)) {
    //Serial.println("SD initialization failed!");
  }
  //Serial.println("initialization done.");
  //if file left from previous
```

```
  if (SD.exists("screen.txt")) {
    SD.remove("screen.txt");
  }
  //fscreen = SD.open("screen.txt", FILE_WRITE);
  if (SD.exists("screen.txt")) {
    Serial.println("screen.txt exists.");
  }
#endif

  //initialize the server
  server.begin();
  Serial.print("server is at ");

  //prints the actual IP, not the one set above
  Serial.println(Ethernet.localIP());
}

void loop() {
#ifdef SD_ENABLE
  screen_parser();
#endif
#ifdef GSM_ENABLE
  gsm_serv();
#endif
  http_serv();
}

/***********************************
```

```
web server()
Initiates an http connection and
handles the HTTP Request
**********************************/
void http_serv(){
  //initialize the client
  EthernetClient client = server.available();


  //if there is no client connected it skips the hole "if" body and
  //jumps at the end of the loop
  if (client) {
    //Serial.println("new client");
    while (client.connected()){
    if (client.available()) {
      //reads one by one the chars of the HTTP Req
      char c = client.read();
      //found the /, begin reading the info
      if(c == '/'){
        //Serial.print("via Ethernet: ");
        //starts decoding the instruction
        decode(client);
        break;
      }
    }
  }
  // give the web browser time to receive the data
  delay(1);
  // close the connection:
```

```
  client.stop();
  //Serial.println("client disonnected");
  }
}

#ifdef GSM_ENABLE
/***********************************************
gsm_serv()
checks if there are unread sms stored at the SIM
***********************************************/
void gsm_serv(){
  if (sms.available() >0){
    Serial.print("via SMS: ");
    decode(sms);
    sms.flush();
  }
}
#endif

/****************
lexical analyzer
***************/
void decode(Stream &stream){
  char c = stream.read();

  //S
  if (c == 'o'){
    //1-o
```

```
    c = stream.read();
    if(c == 'n'){
      //8-on
      Serial.println("ON");
      success_page();
    }else if(c == 'f'){
      //9-of
      c = stream.read();
      if(c == 'f'){
        //10-off
        Serial.println("OFF");
        success_page();
      }else status_page();
    }else status_page();
  }else if(c=='r'){
    //2-r
    c = stream.read();
    if(c=='e'){
      //11-re
      c = stream.read();
      if(c=='s'){
        //12-res
        c = stream.read();
        if(c=='e'){
          //13-rese
          c = stream.read();
          if(c=='t'){
            //14-reset
```

```
            Serial.println("RESET");
            success_page();
          }
        }else status_page();
      }else status_page();
    }else status_page();
  }else if(c=='c'){
    //3-c
    c = stream.read();
    if(c=='o'){
      //15-co
      c = stream.read();
      if(c=='m'){
        //16-com
        c = stream.read();
        if(c=='m'){
          //17-comm
          c = stream.read();
          if(c=='a'){
            //18-comma
            c = stream.read();
            if(c=='n'){
              //19-comman
              c = stream.read();
              if(c=='d'){
                //20-command
                c = stream.read();
                if(c=='s'){
```

```
                //21-commands
                Serial.println("COMMANDS");
                commands_page();
              }else status_page();
            }else status_page();
          }else status_page();
        }else status_page();
      }else status_page();
    }else status_page();
  }else status_page();
}else if(c=='t'){
  //4-t
  c = stream.read();
  if(c=='e'){
    //22-te
    c = stream.read();
    if(c=='m'){
      //23-tem
      c = stream.read();
      if(c=='p'){
        //24-temp
        Serial.println("TEMP");
        success_page();
      }else status_page();
    }else status_page();
  }else status_page();
}else if(c=='l'){
  //5-l
```

```
  c = stream.read();
  if(c=='i'){
    //25-li
    c = stream.read();
    if(c=='g'){
      //26-lig
      c = stream.read();
      if(c=='h'){
        //27-ligh
        c = stream.read();
        if(c=='t'){
          //28-light
          Serial.println("LIGHT");
          success_page();
        }else status_page();
      }else status_page();
    }else status_page();
  }else status_page();
}else if(c=='h'){
  //6-h
  c = stream.read();
  if(c=='u'){
    //29-hu
    c = stream.read();
    if(c=='m'){
      //30-hum
      c = stream.read();
      if(c=='i'){
```

```
          //31-humi
          c = stream.read();
          if(c=='d'){
            //32-humid
            c = stream.read();
            if(c=='i'){
              //33-humidit
              c = stream.read();
              if(c=='y'){
                //34-humidity
                Serial.println("HUMIDITY");
                success_page();
              }else status_page();
            }else status_page();
          }else status_page();
        }else status_page();
      }else status_page();
    }else if(c=='s'){
      //7-s
      c = stream.read();
      if(c=='e'){
        //36-se
        c = stream.read();
        if(c=='n'){
          //37-sen
          c = stream.read();
          if(c=='s'){
```

```
      //38-sens
      c = stream.read();
      if(c=='o'){
        //39-senso
        c = stream.read();
        if(c=='r'){
          //40-sensor
          c = stream.read();
          if(c=='s'){
            //41-sensors
            Serial.println("SENSORS");
            success_page();
          }else status_page();
        }else status_page();
      }else status_page();
    }else status_page();
  }else status_page();
}else if(c=='t'){
  //42-st
  c = stream.read();
  if(c=='a'){
    //43-sta
    c = stream.read();
    if(c=='t'){
      //44-stat
      c = stream.read();
      if(c=='u'){
        //45-statu
```

```
            c = stream.read();
            if(c=='s'){
              //46-status
              Serial.println("STATUS");
              status_page();
            }else status_page();
          }else status_page();
        }else status_page();
      }else status_page();
    }
#ifdef SD_ENABLE
    else if(c=='c'){
      screen_page();
    }
#endif
    else status_page();
  }else if(c=='e'){
    Serial.print("\x1B" "i");
  }else status_page();
}


void status_page(){
  //we should check here if httprequest has ended
  Serial.println("STATUS");
  EthernetClient cl = server.available();
  if(cl){
    boolean currentLineIsBlank = true;
    while (cl.connected()) {
```

```
      char c = cl.read();
      if (c == '\n' && currentLineIsBlank) {
        cl.println("HTTP/1.1 200 OK");
        cl.println("Content-Type: text/html");
        cl.println("Connection: close");
        cl.println();
        cl.println("<!DOCTYPE HTML>");
        cl.println("<html>");
        cl.print("STATUS:");
        //we could print here the instruction set
        cl.println("</html>");
        break;
      }
      if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
      }
      else if (c != '\r') {
        // you've gotten a character on the current line
        currentLineIsBlank = false;
      }
    }
  }
}
void success_page(){
  EthernetClient cl = server.available();
  if(cl){
    boolean currentLineIsBlank = true;
```

```
    while (cl.connected()) {
      char c = cl.read();
      if (c == '\n' && currentLineIsBlank) {
        cl.println("HTTP/1.1 200 OK");
        cl.println("Content-Type: text/html");
        cl.println("Connection: close");
        cl.println();
        cl.println("<!DOCTYPE HTML>");
        cl.println("<html>");
        cl.print("Success");
        //we could print here the instruction set
        cl.println("</html>");
        break;
      }
      if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
      }
      else if (c != '\r') {
        // you've gotten a character on the current line
        currentLineIsBlank = false;
      }
    }
  }
}
void commands_page(){
  EthernetClient cl = server.available();
  if(cl){
```

```
    boolean currentLineIsBlank = true;
    while (cl.connected()) {
      char c = cl.read();
      if (c == '\n' && currentLineIsBlank) {
        cl.println("HTTP/1.1 200 OK");
        cl.println("Content-Type: text/html");
        cl.println("Connection: close");
        cl.println();
        cl.println("<!DOCTYPE HTML>");
        cl.println("<html>");
        cl.println("Commands: (in lower case)");
        cl.println("on\n off\n reset\n commands\n temp\n light\n humidity\n
sensor\n status");
        //we could print here the instruction set
        cl.println("</html>");
        break;
      }
      if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
      }
      else if (c != '\r') {
        // you've gotten a character on the current line
        currentLineIsBlank = false;
      }
    }
  }
}
#ifdef SD_ENABLE
```

```
/************************************
screen_parser()
Inputs: serial stream from screen
and pointer to html file
If two whitespaces in a row ignores it
************************************/
void screen_parser(void)
{
  boolean space;
  char t;
  fscreen = SD.open("screen.txt", FILE_WRITE);
  while(Serial.available()){
    t = Serial.read();
    fscreen.write(t);
/*
    if (Serial.read() == SPACE){
      space = true;
      fscreen.print(' ');
    }
    if ((Serial.read() == SPACE) && space){
      fscreen.print("\n");
    }
*/
  }
  //Serial.println("file printed");
  fscreen.flush();
  fscreen.close();
}
```

```
void screen_page(){
  EthernetClient cl = server.available();
  if(cl){
    boolean currentLineIsBlank = true;
    while (cl.connected()) {
      char c = cl.read();
      if (c == '\n' && currentLineIsBlank) {
        cl.println("HTTP/1.1 200 OK");
        cl.println("Content-Type: text/html");
        cl.println("Connection: close");
        cl.println();
        cl.println("<!DOCTYPE HTML>");
        cl.println("<html>");
        fscreen = SD.open("screen.txt");
        while (fscreen.available()) {
          //Serial.print(".");
          cl.print(fscreen.read());
        }
        cl.println("</html>");
        break;
      }
      if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
      }
      else if (c != '\r') {
        // you've gotten a character on the current line
        currentLineIsBlank = false;
```

```
        }
      }
    }
  }
#endif
```

Appendix B - Description of communication components

B.1 GSM Module description

The M10 is a complete Quad-band GSM/GPRS solution in an LCC type which can be embedded in customer applications, offering the highest reliability and robustness.

Featuring an industry-standard interface and extremely low power consumption, the M10 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor. The M10 can fit into almost all the M2M applications, including VTS, Smart Metering, Wireless POS, Security, etc.

As a part of Quectel's corporate policy of environmental protection, all products comply to the RoHS (Restriction of Hazardous Substances) directive of the European Union (EU Directive 2002/95/EG).

Key benefits:

- Quad-band GSM/GPRS module with a size of 29.0 x 29.0 x 3.6mm
- LCC type suits for customer application
- Embedded powerful internet service protocols, multiple Sockets & IP addresses
- Based on mature and field-proven platform, backed up by our support service, from definition to design and production

B.2 Ethernet Controller Description

The Arduino Ethernet Shield connects Arduino board to the Internet. Simply by plugging this module onto an Arduino board, connect it to network with an RJ45 cable and start controlling any node through the Internet. Every element of the platform – hardware, software and documentation – is freely available and open-source.

Specifications:

- Operating voltage 5V (supplied from the Arduino Board)
- Ethernet Controller: W5100 with internal 16K buffer
- Connection speed: 10/100Mb
- Connection with Arduino on SPI port

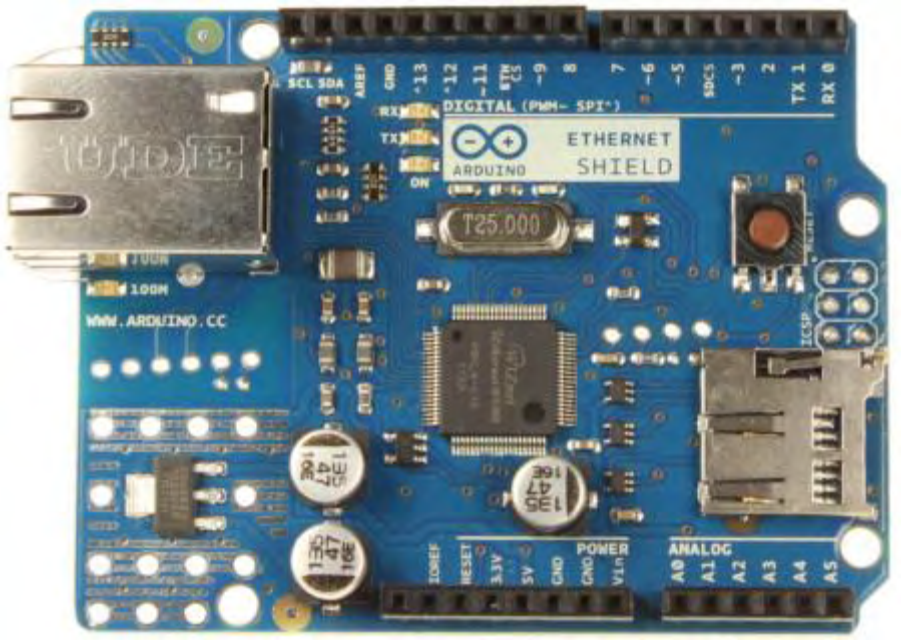


Figure 10 Wiznet Ethernet Module for Arduino board

B.3 MAX232 IC Description

The RS232 serial port protocol (v.24) states -15v to represent binary 1 and +15v to represent binary 0. For TTL communication this is incompatible since TTL uses 0v to represent binary 0 and +5v to represent binary 1. MAX232 chip converts serial signal voltage levels to TTL standards, and also vice versa. It therefore has a transmitter (driver) and a receiver to perform this function.

The support capacitors C4 and C3 are used for the internal voltage inverter that creates the negative voltage level for the serial communication. C1 and C3 are used for the voltage doubler to raise the TTL (5v) level. MAX232 IC is a specialized circuit, which makes standard voltages as required by RS232 standards. This IC is very reliable solution against discharges and short circuits. Also it provides best noise rejection. As it can be seen, there are 2 drivers, and 2 receivers in the MAX232 package. This can be confusing for students and makes the chip look more complicated than it really is, but it's actually very easy since for most applications we generally use only one driver and one receiver. Normally Pins 7, 8, 9, and 10 are used for most of circuits. The other driver and receiver not used could be used as a spare. The diagram below shows the usually schematic of the MAX232 IC circuit. It consists of only 4x 1µF 16V electrolytic capacitor, and the MAX232 IC itself (see Fig. 6.1).

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage generator functions are available as cells in the Texas Instruments Lin ASIC library. The MAX232 is a dual driver/receiver that includes a capacitive voltage.

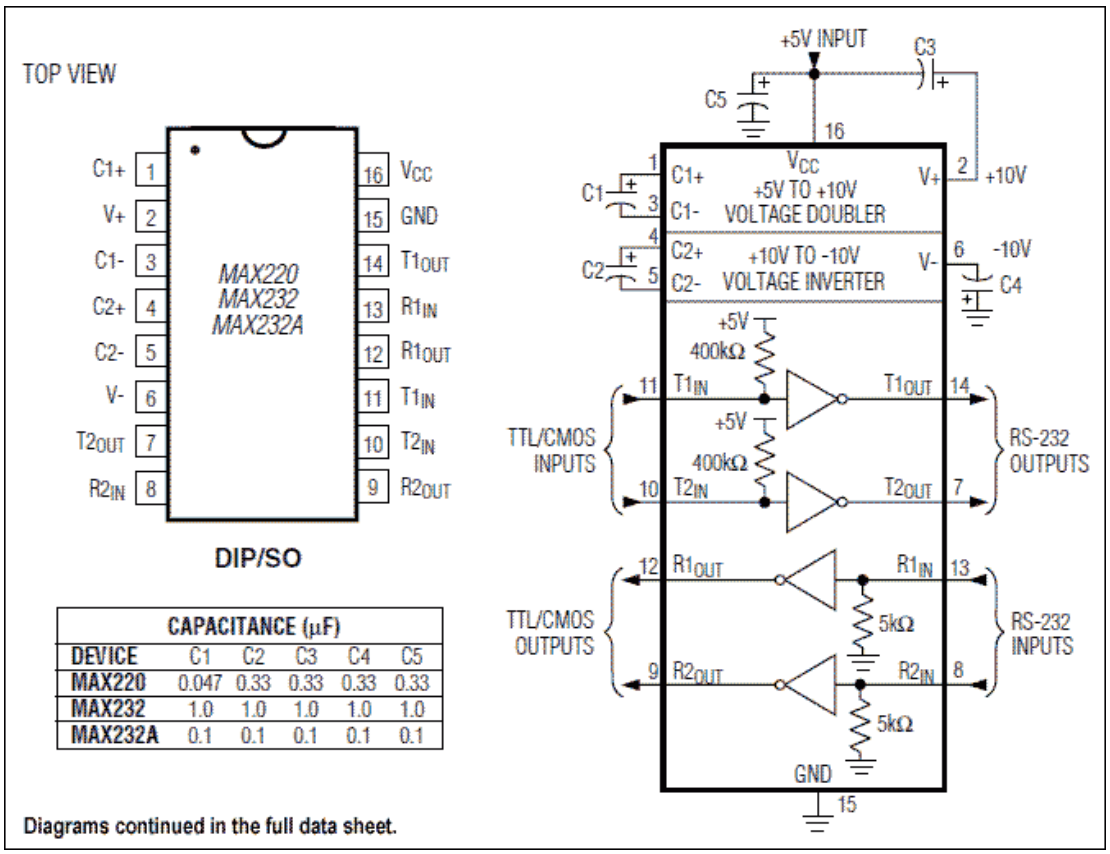
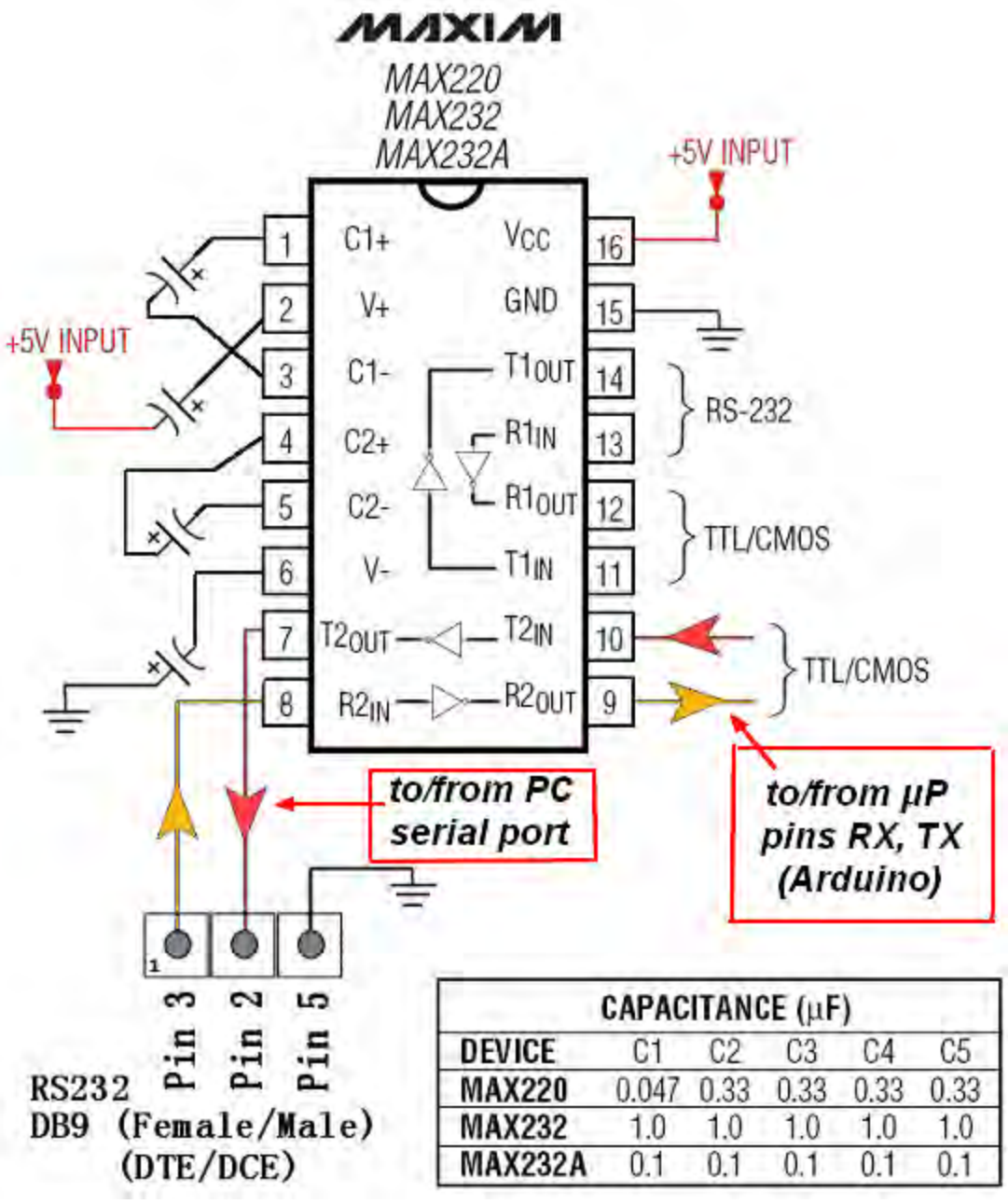


Figure 11 A schematic diagram of the MAX232 IC circuit with 4 electrolytic capacitors of 1µF 16V connected [14]



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 12 A MAX232 IC circuitry allowing a µP to communicate with a PC through its serial port [13], [14]