Syria Arabic Republic
Syrian Private University
Faculty of Computer & Informatics Engineering

Nurse Call System

A Junior Project Presented to the Faculty of Computer Informatics and Engineering
in Partial Fulfillment of the Requirements for the Degree of Bachelor of Engineering in
Communications and Networks

Project prepared by

Mohammad Maamon Al-Shaar

Mohammad Sami Ahmad Al-Ayoubi

Randa Mohammad Hamzeh

Tasnim Kamal Al-Asali

Under the supervision of

Prof.Dr.Eng. Ali Skaf

2014/2015

Nurse Call System

A Junior Project Presented to the Faculty of Computer Informatics and Engineering in Partial Fulfillment of the Requirements for the Degree of Bachelor of Engineering in Communications and Networks

Project prepared by

Mohammad Maamon Al-Shaar

Mohammad Sami Ahmad Al-Ayoubi

Randa Mohammad Hamzeh

Tasnim Kamal Al-Asali

Under the supervision of

Prof.Dr.Eng. Ali Skaf

May 2015

ACKNOWLEDGEMENTS

*Praise be to **ALLAH**, the beneficent the merciful*

*Praise be to **Allah** for the blessing of mind …*

*Praise be to **Allah** for the blessing of love …*

*Praise be to **Allah** for his uncountable blessings, and guidance.*

*To Our homeland **Syria**, the cradle of civilizations…*

*To the **martyrs** who sacrificed their lives for the redemption for homeland…*

*To the **people** who watch over our beloved country's safety and security….*

Deepest gratitude and appreciation to those who contributed to the completion of this work

Dedication

To the source of love and care …
To the person who fears over me …
To the person who prays for me day and night
Mom,

To the teacher of morality …
To the person who gives me power …
To the person who sacrificed everything to get me where I am …
Dad,

To the special people who brings out the best of me …
To the people who stood by me all my life …
To the people who pushed me to be better …
Sisters & Brothers,

To all my family, the symbol of love and giving,
Family,

To the people am proud to have in my life …
Best friends

Along with all hard working and respected
Teachers,

Abstract

In a hospital or any medical center we need to a nurse section that is responsible for patients case. This nurse department supervises a normal patient situation or an emergency situation. In case of emergency the patient needs a way to call nurse immediately, therefore a Nurse Call System was designed to achieve this purpose. This system enables the simultaneous response of a nurse to avoid any deterioration in patient situation.

The designed system is a multiprocessor distributed system composed of a central device destinated to the nurse room with two digit display, and a number of devices for the patients' rooms. Each of the devices is built around an ATmega8 microcontroller.

A multi-drop bus connects all the system puts derived from the inherent UART in the used microcontroller.

Table of Contents

Introduction

A nurse call is a button found around a hospital bed that allows patients in health care settings to alert a nurse or other health care staff member remotely of their need for help. When the button is pressed, a signal alerts staff at the nurse's station, and usually, a nurse or nurse assistant responds to such a call. Some systems also allow the patient to speak directly to the staffer; others simply beep or buzz at the station, requiring a staffer to actually visit the patient's room to determine the patient's needs.

The call button provides the following benefits to patients:

- Enables a patient who is confined to bed and has no other way of communicating with staff to alert a nurse of the need for any type of assistance
- Enables a patient who is able to get out of bed, but for whom this may be hazardous, exhausting, or otherwise difficult to alert a nurse of the need for any type of assistance
- Provides the patient an increased sense of security

The call button can also be used by a health care staff member already with the patient to call for another when such assistance is needed, or by visitors to call for help on behalf of the patient.

Laws in most places require that a call button must be in reach of the patient at all times for example in the patient’s bed or on the table.It is essential to patients in emergencies.There are also laws that vary by location setting the amount of time in which staff must respond to a call.

It is the responsibility of nursing staff to explain to the patients that they have a call button and to teach them how to use it.

The most basic system has nothing more than a button for the patient. When the button is pressed, nursing staff is alerted by a light and/or an audible sound at the nurse's station. This can only be turned off from the patient's bedside, thereby compelling staff to respond to the patient.

Like basic system, wireless type has ability for alerting nursing staff by sound, light or even show messages in a terminal. This type has major advantage that is no wiring works while installation and reducing the costs. It's much more mobile and handy than conventional system.

In some facilities, often in hospitals, a more advanced system is included, in which staff from the nurse's station can communicate directly with patients via intercom. This has the advantage in which staff does not need to waste time walking to the patient's room to determine the reason the patient made the call, and they can determine by speaking to the patient whether the situation is urgent or if it can wait until later.

With the intercom system, the alert can be turned off from the nurse's station, allowing staff to avoid entry into the patient's room if it is determined that the patient's need can be met without doing so.

In this report, a new approach of nurse call system is presented built around microcontrollers. In chapter one we expose the specifications of the proposed system. Chapter 2 deals with the microcontroller’s overview and its architecture. Chapter 3 shows the proposed architecture and the simulated circuit in Proteus. The main practical results are presented in Chapter 4. Finally, a conclusion summarizes the work and shows the future development that could be added to the project.

Chapter 1
Specifications

In the framework of this project, a multiprocessor distributed system is required as shown in figure below. The system can handle up to 100 requests with one priority assigned to the ICU (Intensive Care Unit).

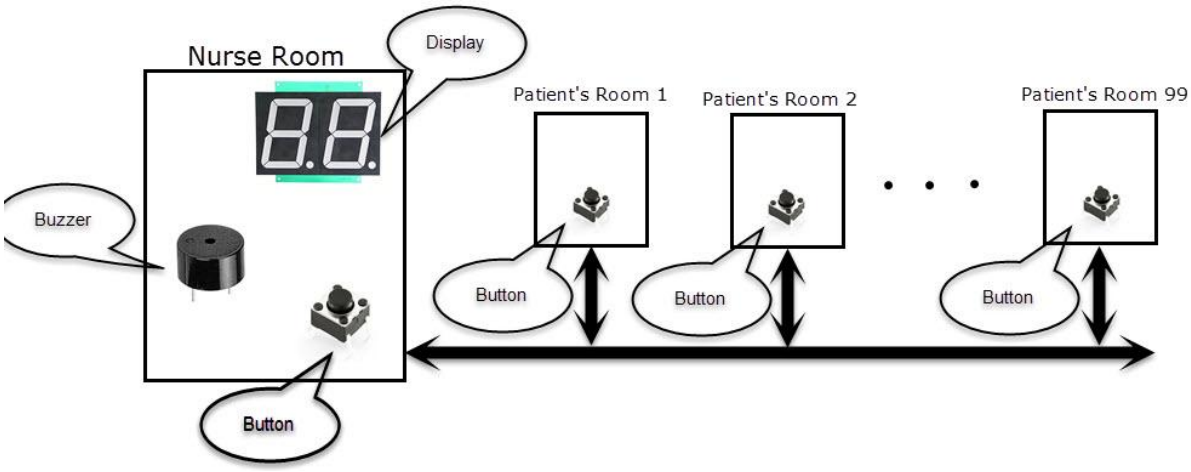


Figure 1.1: Nurse Call System Block Diagram.

The system is composed of:

1. Central unit: at nurse room, equipped with 2 digit to display the room to be served.
2. A number of identical units each destinated to a room.
3. A multi-drop bus to support the data exchanged between the systems puts.

Requests must be memorized in a queue inside the central unit. Pushing the button displays the next request to be served. The ICU request should be treated immediately while the current request is pushed back in the queue.

Chapter 2
Theoretical Study

2.1 Microcontrollers

Like all good things, this powerful component is basically very simple. It is made by mixing tested and high- quality "ingredients" (components) as per following receipt:

1. The simplest computer processor is used as the "brain" of the future system.
2. Depending on the taste of the manufacturer, a bit of memory, a few A/D converters, timers, input/output lines etc. are added
3. All that is placed in some of the standard packages.
4. A simple software able to control it all and which everyone can easily learn about has been developed.

On the basis of these rules, numerous types of microcontrollers were designed and they quickly became man's invisible companion. Their incredible simplicity and flexibility conquered us a long time ago and if you try to invent something about them, you should know that you are probably late, someone before you has either done it or at least has tried to do it [1].

The following things have had a crucial influence on development and success of the microcontrollers:

- Powerful and carefully chosen electronics embedded in the microcontrollers can independently or via input/output devices (switches, push buttons, sensors, LCD displays, relays etc.), control various processes and devices such as industrial automation, electric current, temperature, engine performance etc.
- Very low prices enable them to be embedded in such devices in which, until recent time it was not worthwhile to embed anything. Thanks to that, the world is overwhelmed today with cheap automatic devices and various “smart” appliances.
- Prior knowledge is hardly needed for programming. It is sufficient to have a PC (software in use is not demanding at all and is easy to learn) and a simple device (called the programmer) used for “loading” ready-to-use programs into the microcontroller.

2.1.1 How does the microcontroller operate?

Even though there is a large number of different types of microcontrollers and even more programs created for their use only, all of them have many things in common. Thus, if you learn to handle one of them you will be able to handle them all. A typical scenario on the basis of which it all functions is as follows:

1. Power supply is turned off and everything is still…the program is loaded into the microcontroller, nothing indicates what is about to come…
2. Power supply is turned on and everything starts to happen at high speed! The control logic unit keeps everything under control. It disables all other circuits except quartz crystal to operate. While the preparations are in progress, the first milliseconds go by.
3. Power supply voltage reaches its maximum and oscillator frequency becomes stable. SFRs are being filled with bits reflecting the state of all circuits within the microcontroller. All pins are configured as inputs. The overall electronics starts operation in rhythm with pulse sequence. From now on the time is measured in micro and nanoseconds.
4. Program Counter is set to zero. Instruction from that address is sent to instruction decoder which recognizes it, after which it is executed with immediate effect.
5. The value of the Program Counter is incremented by 1 and the whole process is repeated...several million times per second.

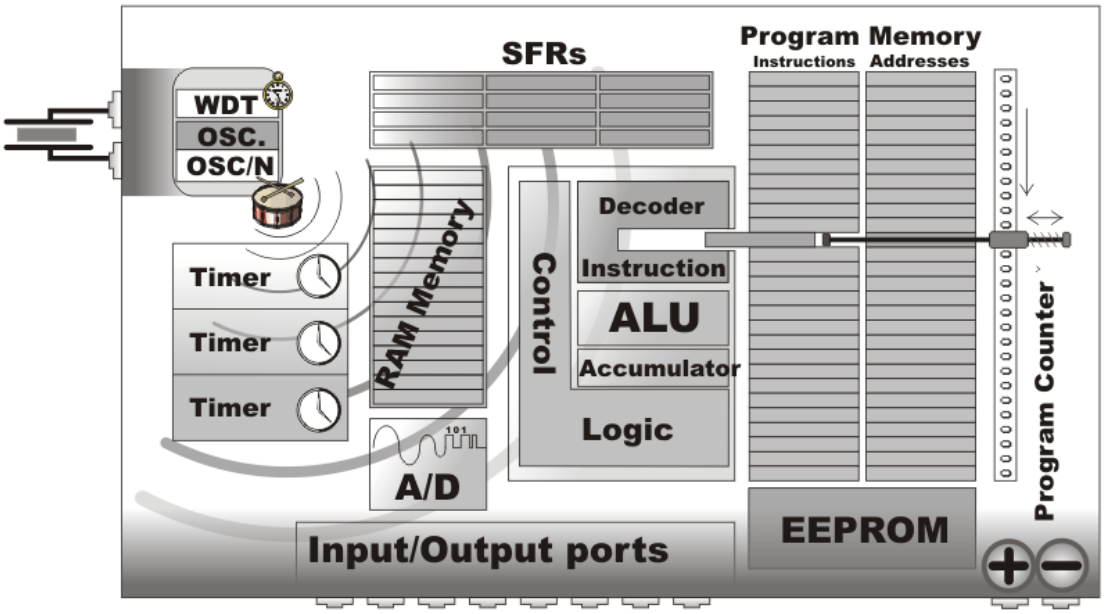


Figure 2.1: microcontroller general component.

2.1.2 Inside microcontrollers

As you can see, all the operations within the microcontroller are performed at high speed and quite simply, but the microcontroller itself would not be so useful if there are not special circuits which make it complete.

Read Only Memory (ROM)

Read Only Memory (ROM) is a type of memory used to permanently save the program being executed. The size of the program that can be written depends on the size of this memory. ROM can be built in the microcontroller or added as an external chip, which depends on the type of the microcontroller. Both options have some disadvantages. If ROM is added as an external chip, the microcontroller is cheaper and the program can be considerably longer. At the same time, a number of available pins is reduced as the microcontroller uses its own input/output ports for connection to the chip. The internal ROM is usually smaller and more expensive, but leaves more pins available for connecting to peripheral environment. The size of ROM ranges from 512B to 64KB [1].

Random Access Memory (RAM)

Random Access Memory (RAM) is a type of memory used for temporary storing data and intermediate results created and used during the operation of the microcontrollers. The content of this memory is cleared once the power supply is off. For example, if the program performs an addition, it is necessary to have a register standing for what in

everyday life is called the “sum”. For that purpose, one of the registers in RAM is called the "sum" and used for storing results of addition. The size of RAM goes up to a few KBs.

Electrically Erasable Programmable ROM (EEPROM)

The EEPROM is a special type of memory not contained in all microcontrollers. Its contents may be changed during program execution (similar to RAM), but remains permanently saved even after the loss of power (similar to ROM). It is often used to store values, created and used during operation (such as calibration values, codes, values to count up to etc.), which must be saved after turning the power supply off. A disadvantage of this memory is that the process of programming is relatively slow. It is measured in milliseconds.

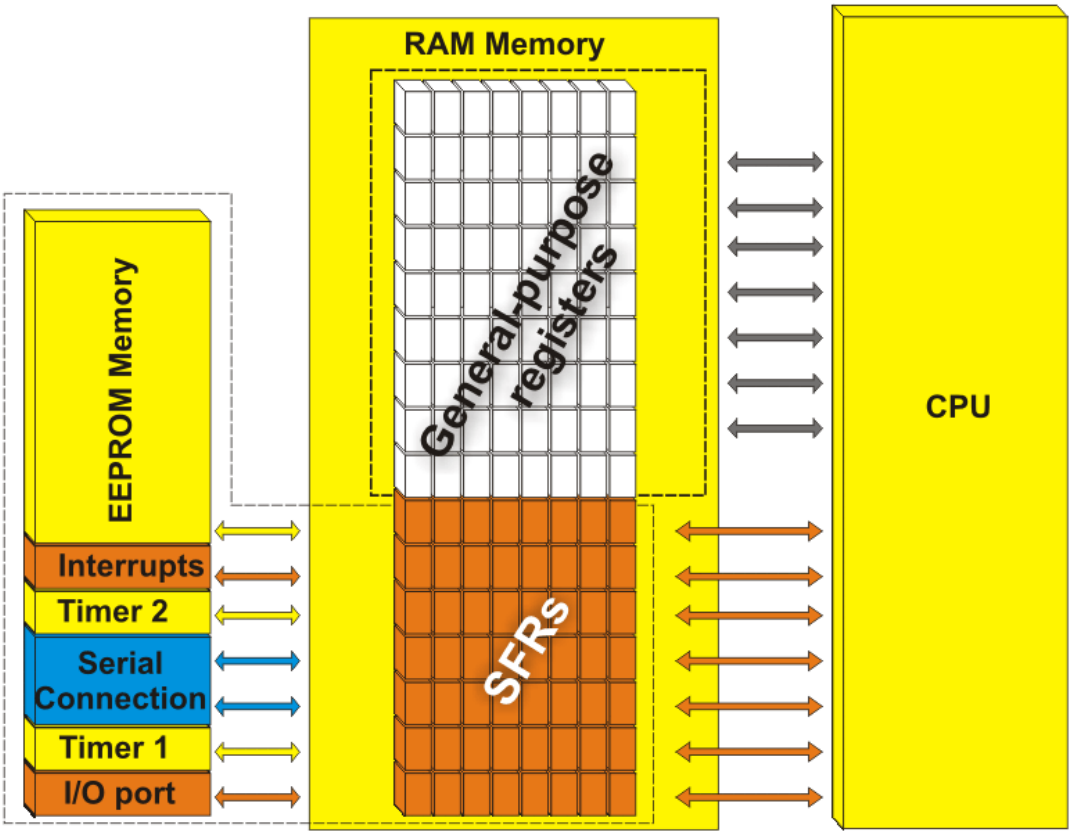


Figure 2.2: Memory architecture of Microcontroller.

Special Function Registers (SFR)

Special function registers are part of RAM memory. Their purpose is predefined by the manufacturer and cannot be changed therefore. Since their bits are physically connected to particular circuits within the microcontroller, such as A/D converter, serial communication module etc., any change of their state directly affects the operation of the microcontroller

or some of the circuits. For example, writing zero or one to the SFR controlling an input/output port causes the appropriate port pin to be configured as input or output. In other words, each bit of this register controls the function of one single pin.

Program Counter

Program Counter is an engine running the program and points to the memory address containing the next instruction to execute. After each instruction execution, the value of the counter is incremented by 1. For this reason, the program executes only one instruction at a time just as it is written. However…the value of the program counter can be changed at any moment, which causes a "jump" to a new memory location. This is how subroutines and branch instructions are executed. After jumping, the counter resumes even and monotonous automatic counting +1, +1, +1…

Central Processor Unit (CPU)

As its name suggests, this is a unit which monitors and controls all processes within the microcontroller and the user cannot affect its work [1]. It consists of several smaller subunits, of which the most important are:

- ***Instruction decoder*** is a part of the electronics which recognizes program instructions and runs other circuits on the basis of that. The abilities of this circuit are expressed in the "instruction set" which is different for each microcontroller family.
- ***Arithmetical Logical Unit (ALU)*** performs all mathematical and logical operations upon data.
- ***Accumulator*** is an SFR closely related to the operation of ALU. It is a kind of working desk used for storing all data upon which some operations should be executed (addition, shift etc.). It also stores the results ready for use in further processing. One of the SFRs, called the Status Register, is closely related to the accumulator, showing at any given time the "status" of a number stored in the accumulator (the number is greater or less than zero etc.).

Input/output ports (I/O Ports)

In order to make the microcontroller useful, it is necessary to connect it to peripheral devices. Each microcontroller has one or more registers (called a port) connected to the microcontroller pins.

Why do we call them input/output ports? Because it is possible to change a pin function according to the user's needs. These registers are the only registers in the microcontroller the state of which can be checked by voltmeter!

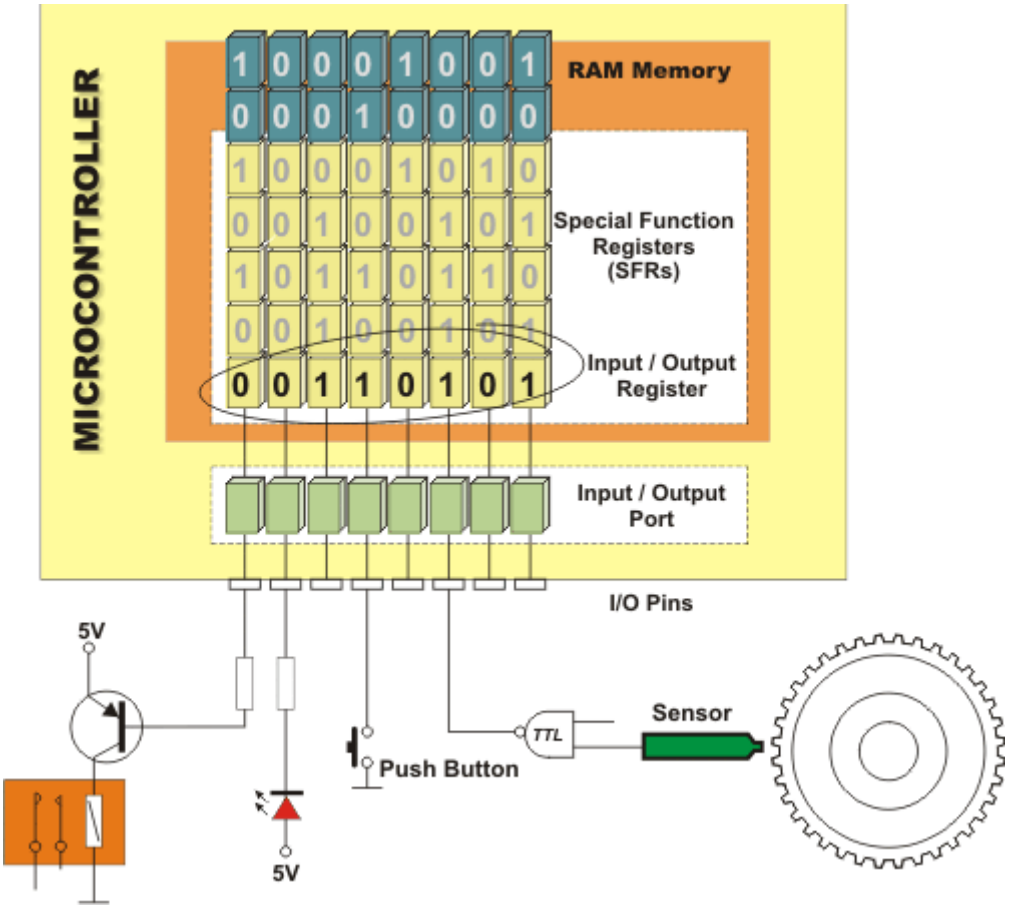


Figure 2.3: Input/Output Ports.

Oscillator

Even pulses generated by the oscillator enable harmonic and synchronous operation of all circuits within the microcontroller. It is usually configured as to use quartz-crystal or ceramics resonator for frequency stabilization. It can also operate without elements for frequency stabilization (like RC oscillator). It is important to say that program instructions are not executed at the rate imposed by the oscillator itself, but several times slower. It happens because each instruction is executed in several steps. For some microcontrollers, the same number of cycles is needed to execute any instruction, while it's different for other microcontrollers. Accordingly, if the system uses quartz crystal with a frequency of 20MHz, the execution time of an instruction is not expected 50nS, but 200, 400 or even 800 ns, depending on the type of the microcontroller!

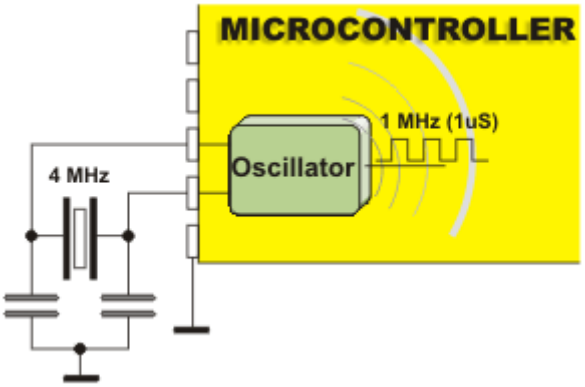


Figure 2.4: Microcontroller Oscillator.

Timers/Counters

Most programs use these miniature electronic "stopwatches" in their operation. These are commonly 8- or 16-bit SFRs the contents of which is automatically incremented by each coming pulse. Once the register is completely loaded, an interrupt is generated!

If these registers use an internal quartz oscillator as a clock source, then it is possible to measure the time between two events (if the register value is T1 at the moment measurement has started, and T2 at the moment it has finished, then the elapsed time is equal to the result of subtraction T2-T1). If the registers use pulses coming from external source, then such a timer is turned into a counter [1].

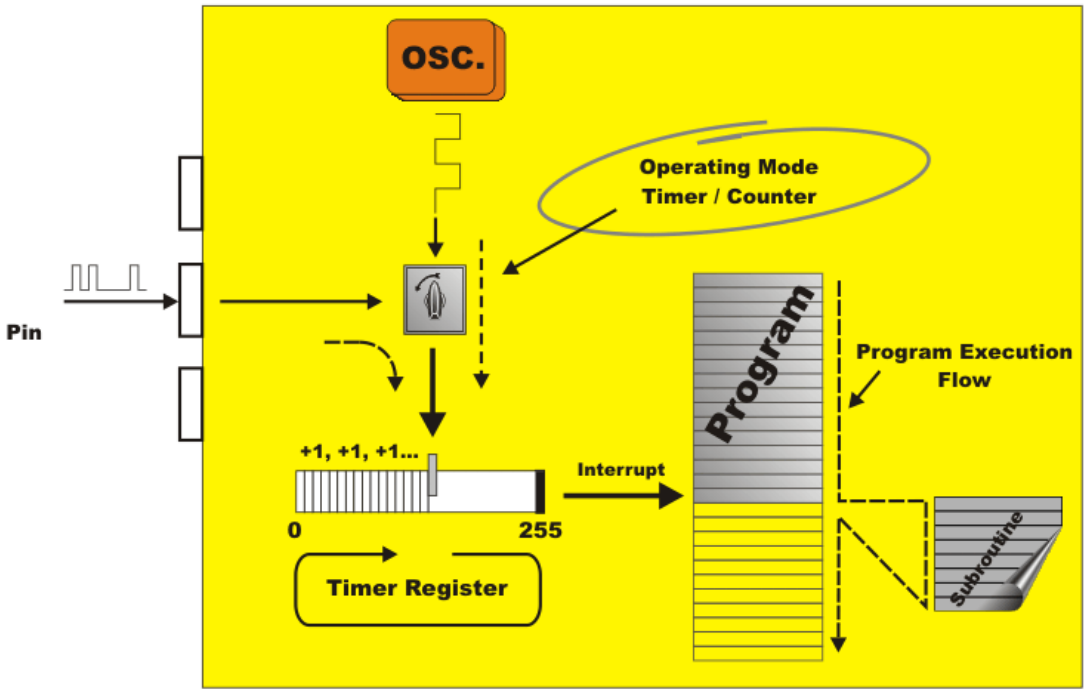


Figure 2.5: Microcontroller Timers/Counters.

Watchdog timer

The Watchdog Timer is a timer connected to a completely separate RC oscillator within the microcontroller.

If the watchdog timer is enabled, every time it counts up to the program end, the microcontroller reset occurs and program execution starts from the first instruction. The point is to prevent this from happening by using a special command. The whole idea is based on the fact that every program is executed in several longer or shorter loops.

If instructions resetting the watchdog timer are set at the appropriate program locations, besides commands being regularly executed, then the operation of the watchdog timer will not affect the program execution.

If for any reason (usually electrical noise in industry), the program counter "gets stuck" at some memory location from which there is no return, the watchdog will not be cleared, so the register's value being constantly incremented will reach the maximum et voila! Reset occurs!

Power Supply Circuit

There are two things worth attention concerning the microcontroller power supply circuit:

Brown out is a potentially dangerous state which occurs at the moment the microcontroller is being turned off or when power supply voltage drops to the lowest level due to electric noise. As the microcontroller consists of several circuits which have different operating voltage levels, this can cause its out of control performance. In order to prevent it, the microcontroller usually has a circuit for brown out reset built-in. This circuit immediately resets the whole electronics when the voltage level drops below the lower limit.

Reset pin is usually referred to as Master Clear Reset (*MCLR*) and serves for external reset of the microcontroller by applying logic zero (0) or one (1) depending on the type of the microcontroller. In case the brown out is not built in the microcontroller, a simple external circuit for brown out reset can be connected to this pin [1].

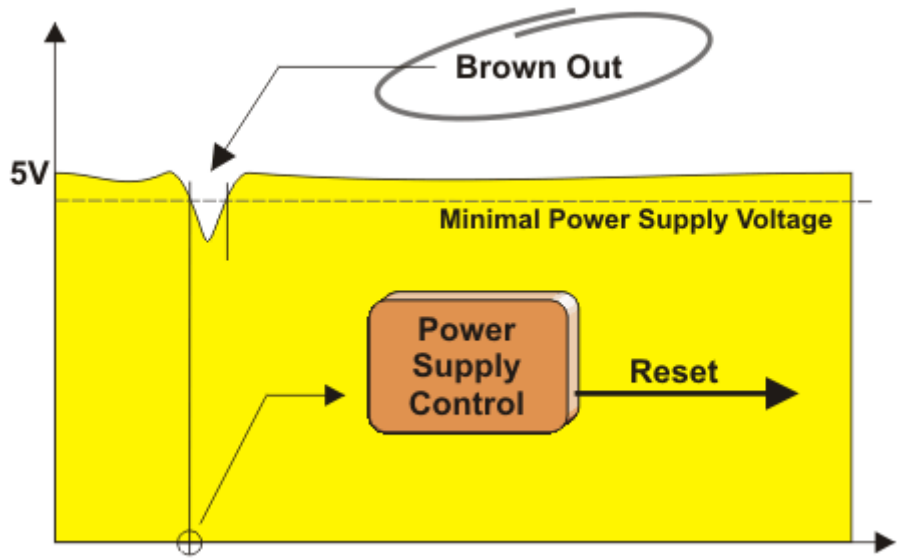


Figure 2.6: Microcontroller Power Supply.

Serial communication

Parallel connections between the microcontroller and peripherals established over I/O ports are the ideal solution for shorter distances up to several meters. However, in other cases, when it is necessary to establish communication between two devices on longer distances it is obviously not possible to use parallel connections. Then, serial communication is the best solution.

Today, most microcontrollers have several different systems for serial communication built in as a standard equipment. Which of them will be used depends on many factors of which the most important are:

- How many devices the microcontroller has to exchange data with?
- How fast the data exchange has to be?
- What is the distance between devices?
- Is it necessary to send and receive data simultaneously?

One of the most important things concerning serial communication is the Protocol which should be strictly observed. It is a set of rules which must be applied in order that devices can correctly interpret data they mutually exchange. Fortunately, the microcontrollers automatically take care of this, so the work of the programmer/user is reduced to a simple write (data to be sent) and read (received data).

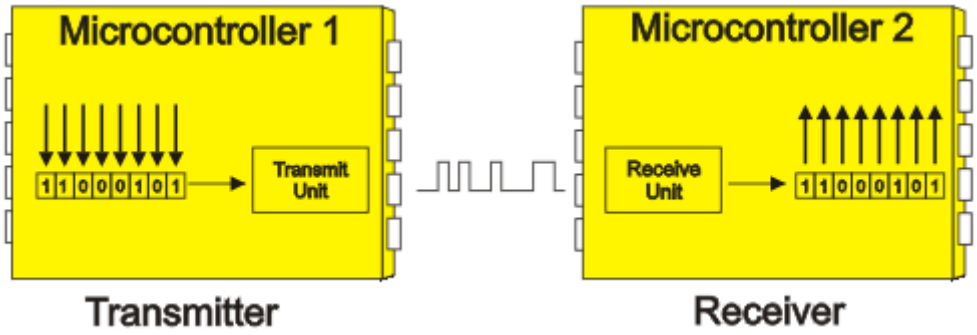


Figure 2.7: Microcontroller UART.

Program

Unlike other integrated circuits which only need to be connected to other components and turn the power supply on, the microcontrollers need to be programmed first. This is a so called "bitter pill" and the main reason why hardware-oriented electronics engineers stay away from microcontrollers. It is a trap causing huge losses because the process of programming the microcontroller is basically very simple.

In order to write a program for the microcontroller, several "low-level" programming languages can be used such as Assembly, C and Basic (and their versions as well). Writing program procedure consists of simple writing instructions in the order in which they should be executed. There are also many programs running in Windows environment used to facilitate the work providing additional visual tools [1].

Chapter 3
Proposed Architecture

3.1 Proteus Design Suite

The Proteus Design Suite is wholly unique in offering the ability to co-simulate both high and low-level micro-controller code in the context of a mixed-mode SPICE circuit simulation. With this Virtual System Modelling facility, you can transform your product design cycle, reaping huge rewards in terms of reduced time to market and lower costs of development.

If one person designs both the hardware and the software then that person benefits as the hardware design may be changed just as easily as the software design. In larger organizations where the two roles are separated, the software designers can begin work as soon as the schematic is completed; there is no need for them to wait until a physical prototype exists.

In short, Proteus VSM improves efficiency, quality and flexibility throughout the design process.

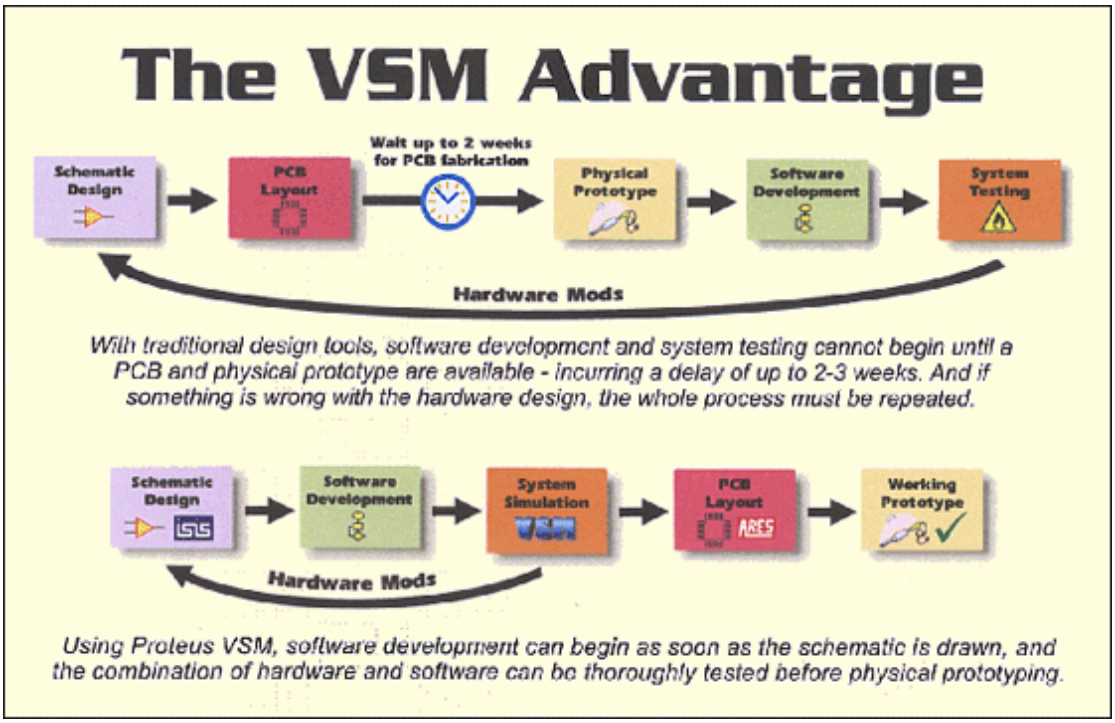


Figure 3.1: Proteus Software Overview.

3.1.1 What is Proteus VSM?

Proteus Virtual System Modelling (VSM) combines mixed mode SPICE circuit simulation, animated components and microprocessor models to facilitate co-simulation of complete microcontroller based designs. For the first time ever, it is possible to develop and test such designs before a physical prototype is constructed.

This is possible because you can interact with the design using on screen indicators such as LED and LCD displays and actuators such as switches and buttons. The simulation takes place in real time (or near enough to it): a 1GMHz Pentium III can simulate a basic 8051 system clocking at over 12MHz. Proteus VSM also provides extensive debugging facilities including breakpoints, single stepping and variable display for both assembly code and high level language source.

3.2 Project Schematic

3.2.1 General Project Schematic

We used Proteus VSM to simulate Nurse Call System and show the results. In the figure below we can see the general diagram which consists of 1 master unit that will fixed in the nurse department, while the other 3 units are a slave units will put near the patient's bed.

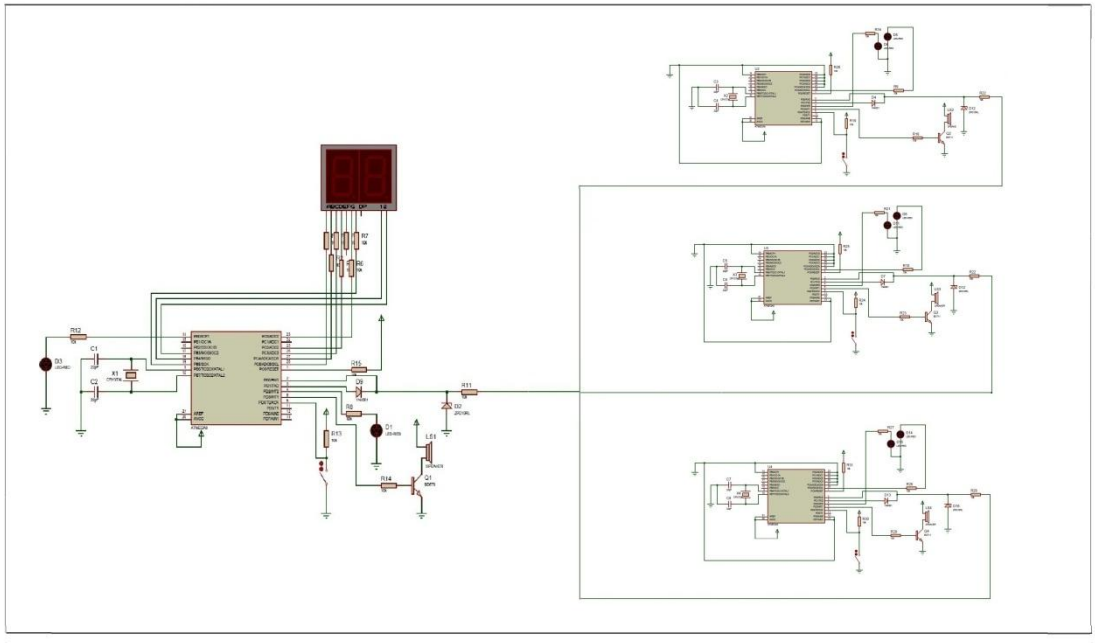


Figure 3.2: Overall Project Schematic.

3.2.2 Master Unit Peripherals

As we mention above, Master unit will fix in the Nurse Department. As we can see in the figure below, the microcontroller deals with Seven Segment Display that shows the patient's room number, in addition to a buzzer or speaker alarm that starts as patient's call button pressed.

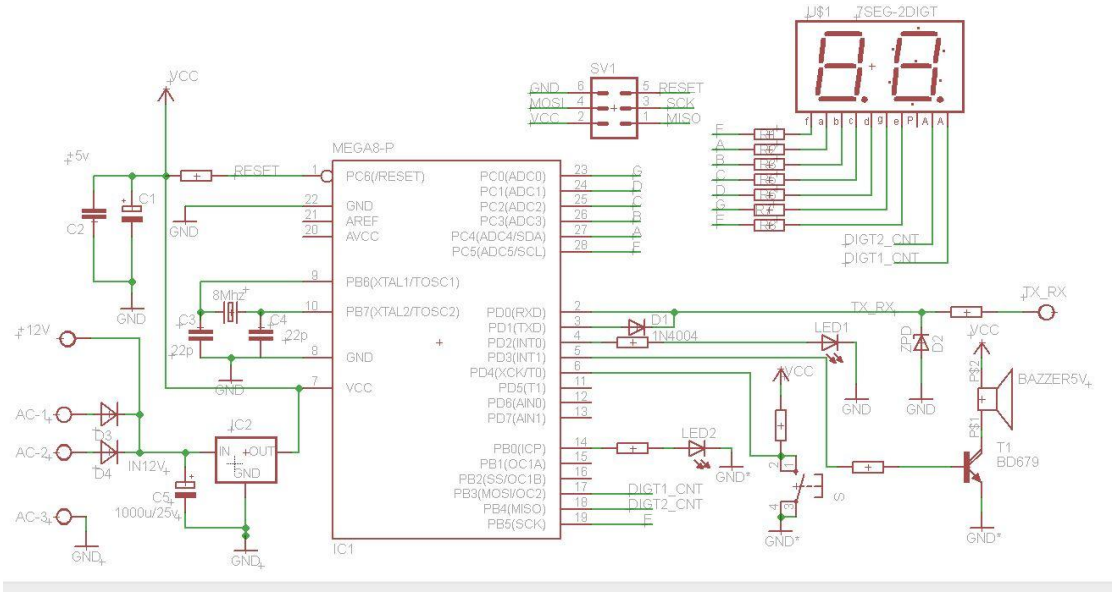


Figure 3.3: Master Unit Schematic.

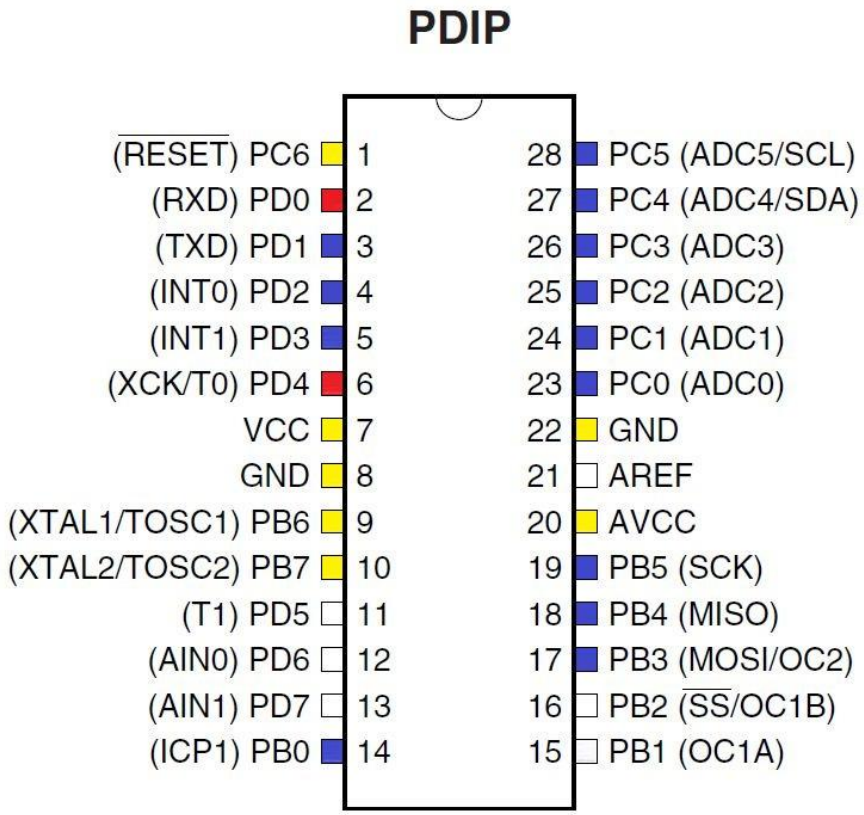


Figure 3.4: Master Unit Microcontroller peripherals.

3.2.3 Slave Unit Peripherals

Slave unit responsible is to send a request to Master Unit and receive acknowledgements from it. It contains a button that will fix near to patient's bed, when the button was pressed, a request sends to Master Unit tells it to start the buzzer in the Nurse Department.

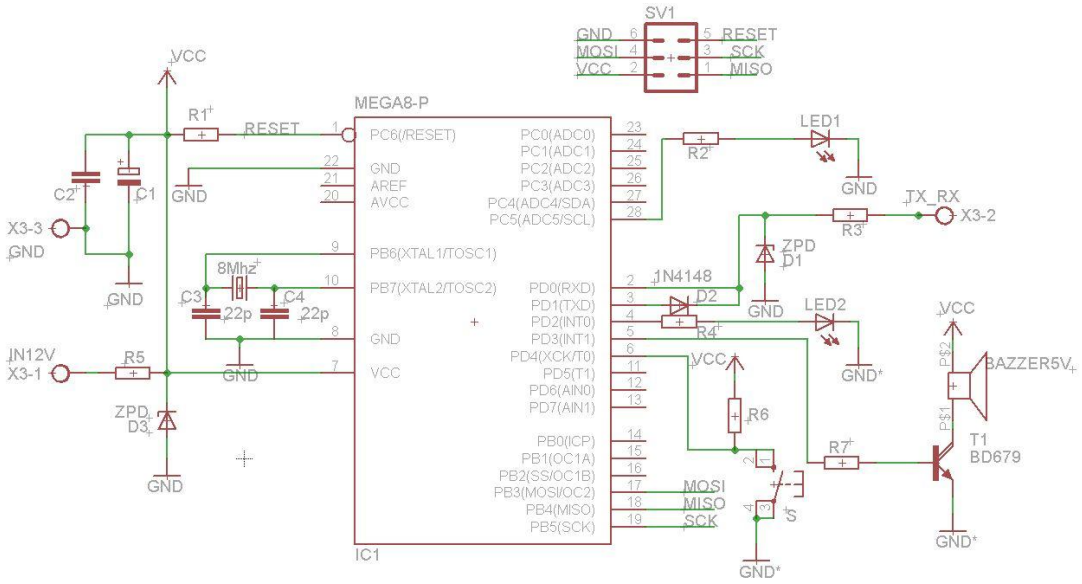


Figure 3.5: Slave Unit Schematic.

3.2.4 Power Supply Unit

The power supply was designed to convert 220V AC to 12V DC and 5V DC, that used to supplied the other units with the corresponding voltage.

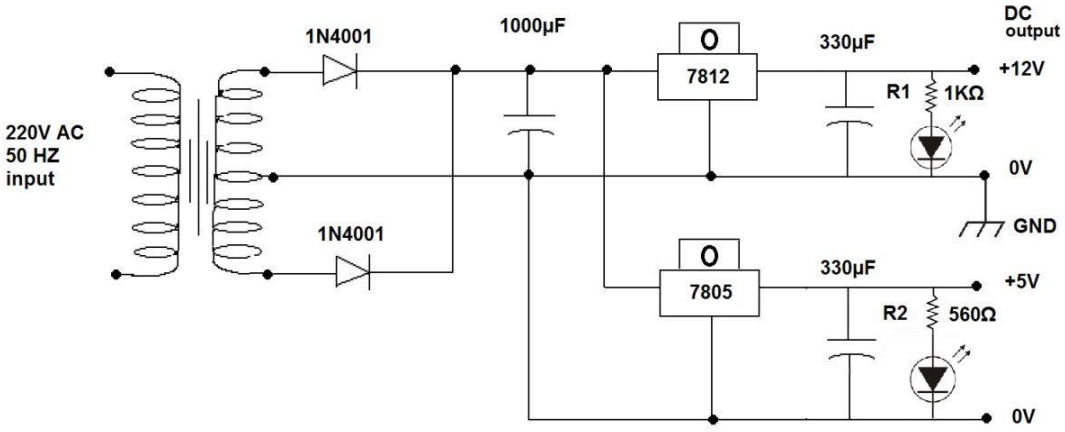


Figure 3.6: Power Supply Schematic.

Chapter 4
Practical Implementation

4.1 Introduction

The programming language used to program the microcontroller is C, which can be written in Code Vision AVR software that deals with the microcontrollers. The micro that we used is ATmega8. The projects consists of two units “Master and Slaves”. All units connected with each other through (+12, GND and DATA). The DATA lines connected with UART ports in the microcontrollers using a special continuation explained below.

The Master Unit is a central unit to be fixed in Nurse Department and we made 1 Master PCB board. It consists of microcontroller, Digital display with 2 7-Segments, LED, Buzzer and Button.

The Slave Unit is a terminal unit that will be in patients’ rooms. We made 3 PCBs with one of them has higher priority to test the functionality of the system. Each contains a microcontroller, LED1, LED2, Button and DIP Switch to specify the room number.

We can serve up to 100 room. We will give room “Ur” higher priority in request execution. Address 00 is assigned to this room.

4.2 Multi drop bus

A multi-drop bus (MDB) is a computer bus in which all components are connected to the electrical circuit. A process of arbitration determines which device sends information at any point. The other devices listen for the data they are intended to receive.

Multi-drop buses have the advantage of simplicity and extensibility. However, modern SDRAM chips exemplify the problem of electrical impedance discontinuity. Fully Buffered DIMM is an alternative approach to connecting multiple DRAM modules to a memory controller. Since 2000, multi-drop standards such as PCI and Parallel ATA are increasingly being replaced by point-to-point systems such as PCI Express and SATA.

In our case, we modified the UART connection (RXD, TXD) to form a multidrop bus with one data wire only as shown in figure below.

Normally, TX is high, when one microcontroller wants to send a value to the bus, zeroes are based as the diode is normally biased. The other TX pins on the other devices are unaffected of their diodes are reverse biased.

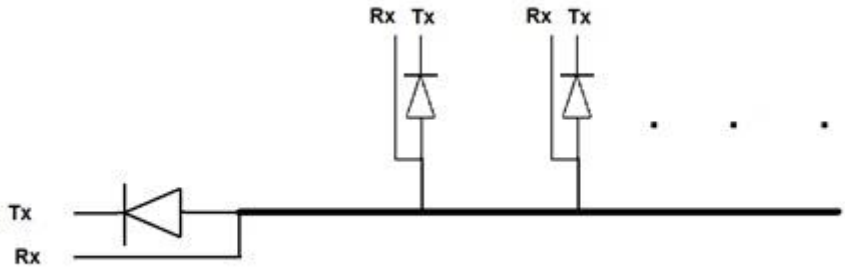


Figure 4.1: Multidrop Bus Overview.

Furthermore, this connection allows us to verify the correct transmission of data as the transmitter also receives the sent data. If the received data is different from the sent, then the transmitter knows and can resend the data again as explained below.

4.2 Slave Unit

Slave unit composed of external Crystal Oscillator which ensure the stability of clocking. We can see also the pins for programming the microcontroller.

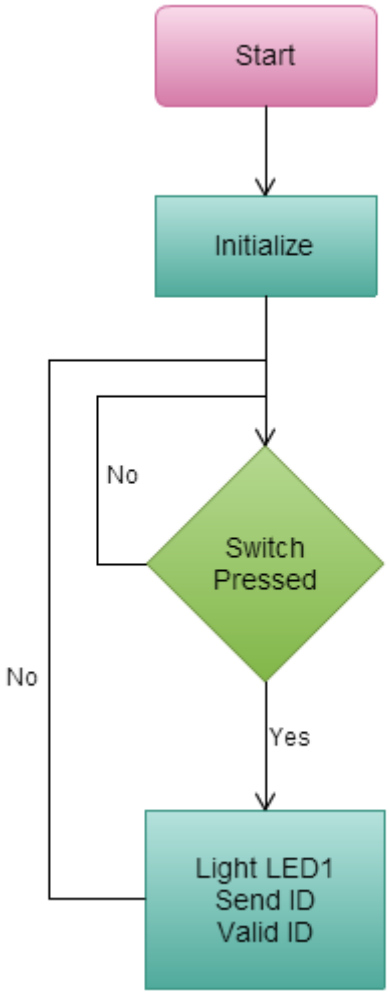


Figure 4.2: Slave Unit Flow Chart.

The request will be sent when the button is pressed, which will execute the following:

LED1 will be lit, which indicate that the request is being sent now. The transmission will be done through (TXD) of the microcontroller. The microcontroller will send the ID of patient's room which will be identified using DIP Switch connected to the microcontroller. After that, microcontroller will receive the data from RXD and compare the ID with the one it has, in case of not equal the microcontroller will wait for a period related to its ID, then it will resend ID automatically through UART and compare the receive data with ID … etc. On the other hand, when the received ID equal to the microcontroller ID, the microcontroller will stop sending the ID to the master unit.

LED1 stays lit until the microcontroller receives a signal to switch off the LED1, which tells the slave unit that its request processing now.

LED2 function is to toggle on and off every 2 seconds which indicates that the device is in working state and it is ready for sending request in any time.

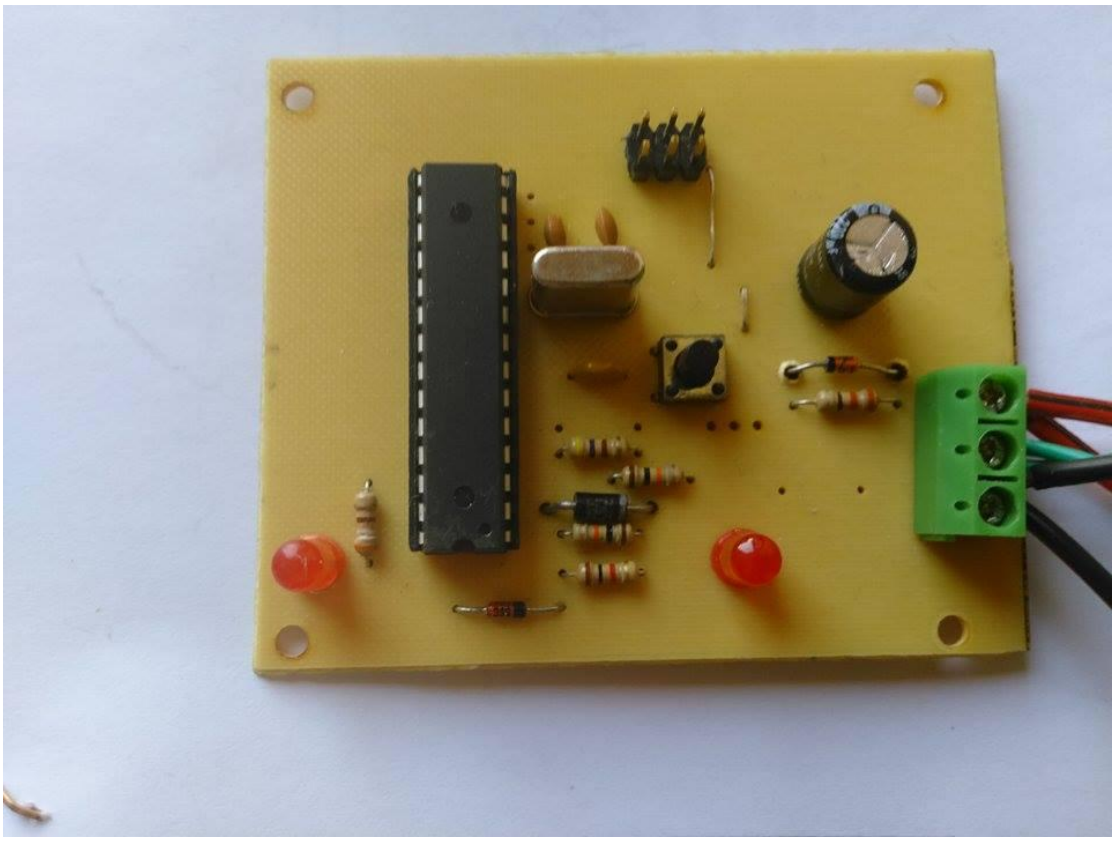


Figure 4.3: Slave Unit Practical Circuit.

4.3 Master Unit

Master unit contains external crystal oscillator, which puts for the same reason. We use a transistor that operates the buzzer.

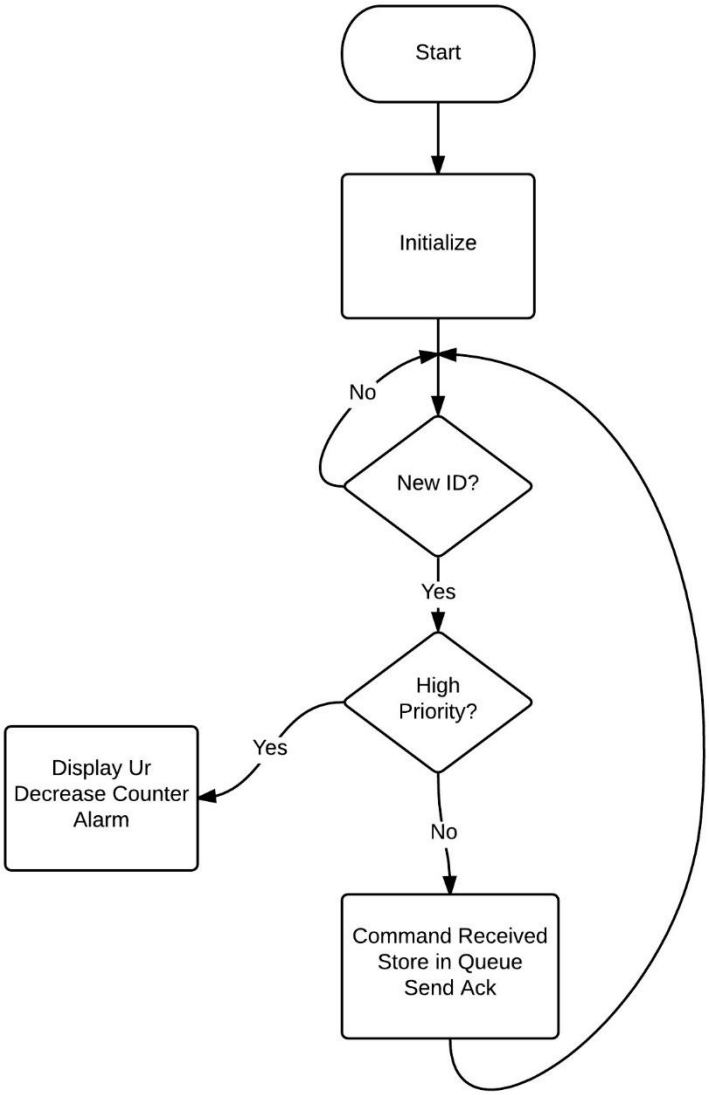


Figure 4.4: Master Unit Flow chart.

When requests arrived at the master microcontroller, it will verify that the received ID is correct. After that, it will save the ID in queue to process the request in order except in case of room “Ur” address 00.

At any request arrival, the microcontroller will start a buzzer and LED. In addition, it will show an indicator at the display indicates that a request has to be executed. When the nurse press the button, the microcontroller will DE queue the room number and show at the display

to send one of the nurse to the corresponding room. When the request done, the nurse will press again the button to see if any new order had been request.

This button will do two function: first it will send a signal to the corresponding slave unit to light off the LED1 which indicate that the request was executed and there is a nurse in its way to the room. Second, the microcontroller will dequeue the next order and shows at the display to be executed or it will show “EE” at display which indicates that there is no order to be executed.

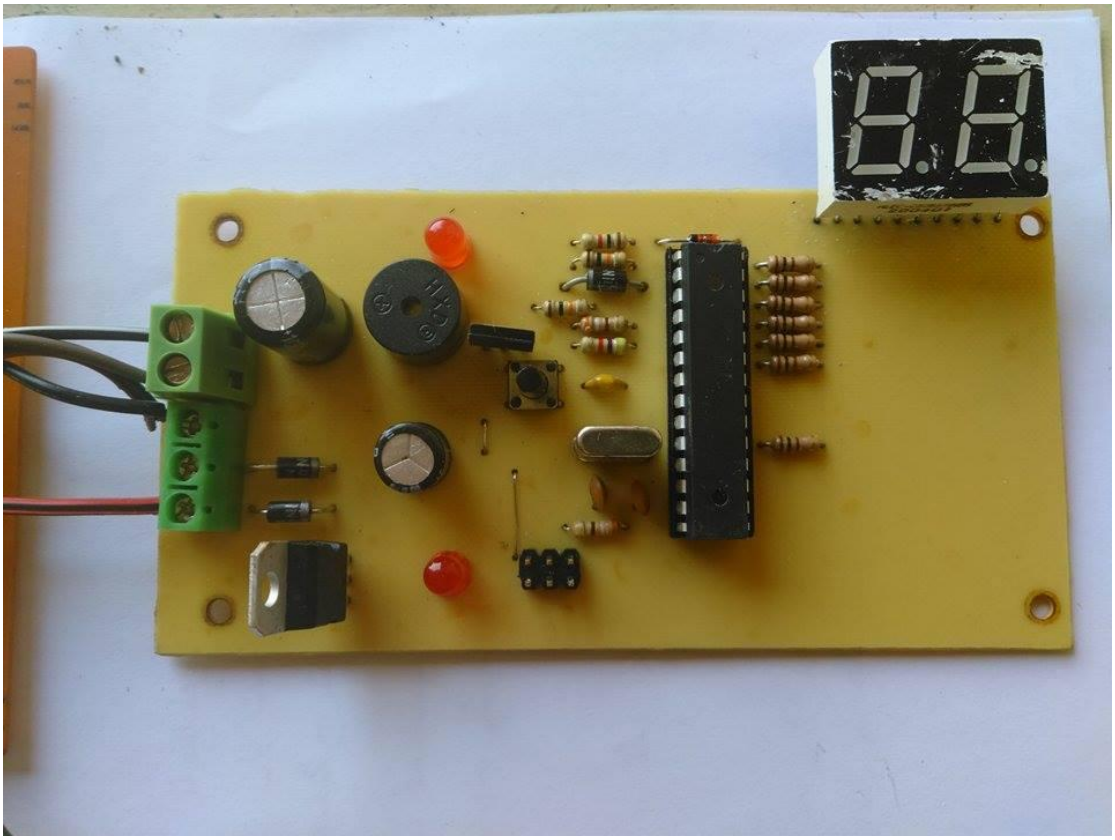


Figure 4.5: Master Unit Practical Circuit.

4.4 Practical Execution

In execution we receive the following requests: (23, 56, and “Ur”). The requests will be executed in the following order (“Ur”, 23, and 56).

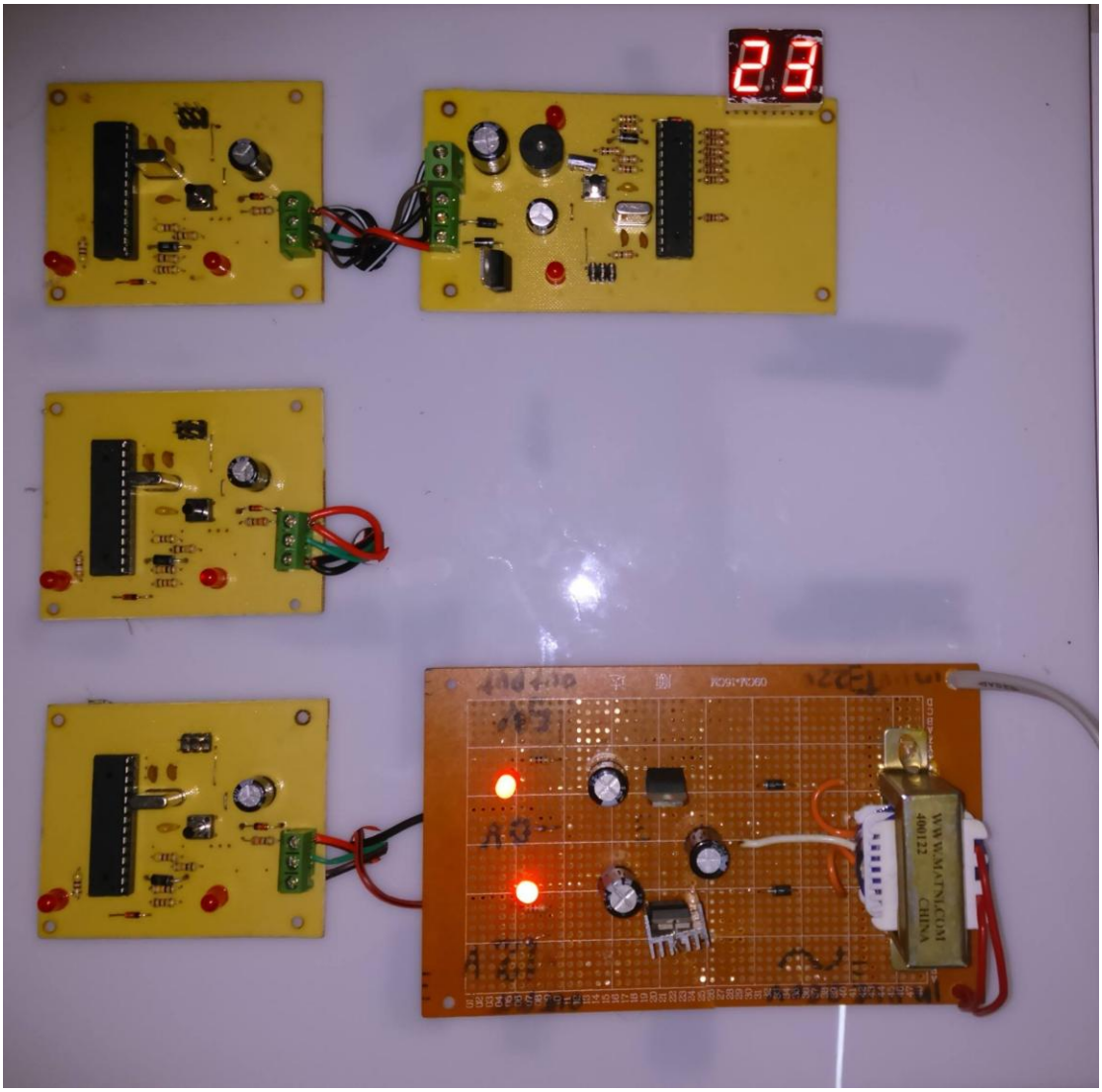


Figure 4.6: Practical Circuits.

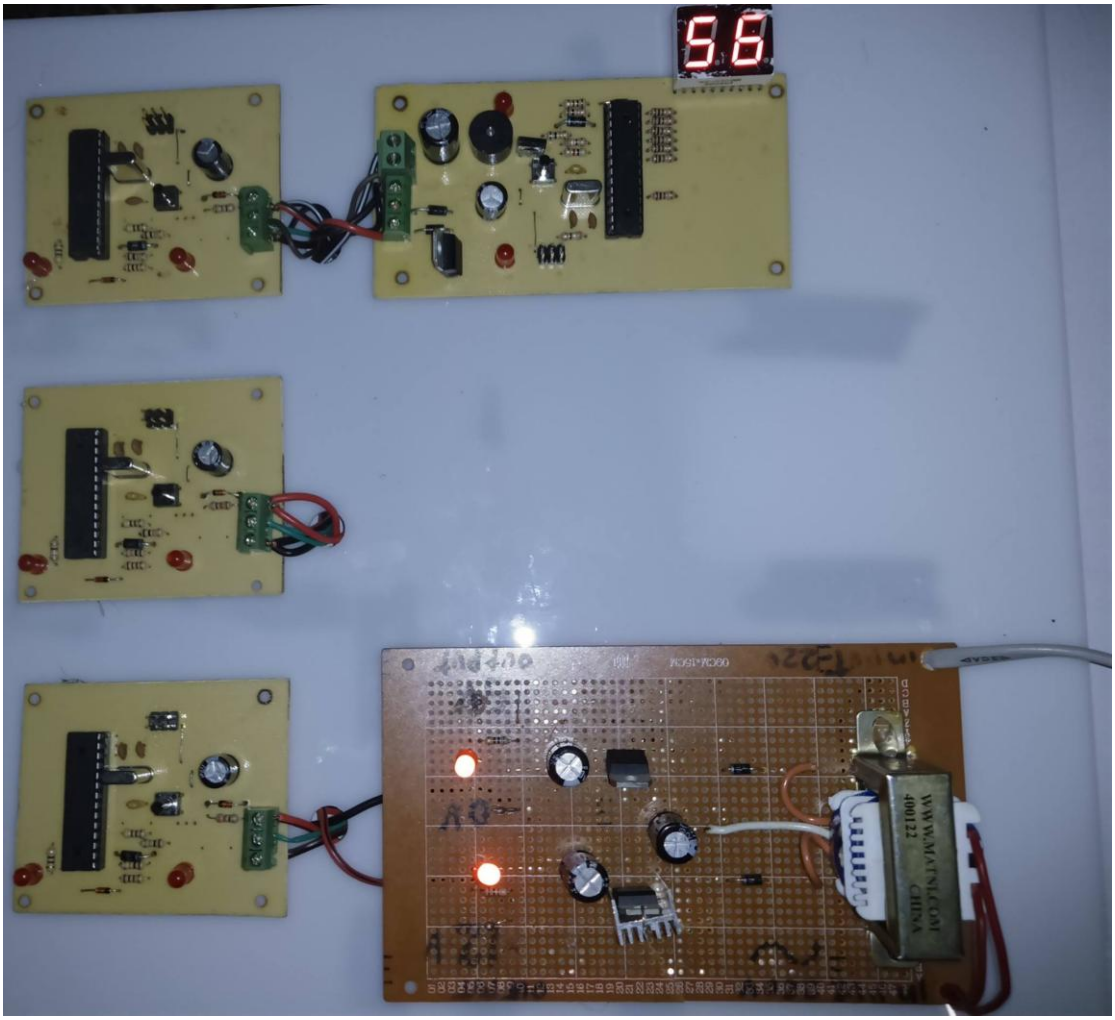


Figure 4.7: Room 56 requests a Nurse.

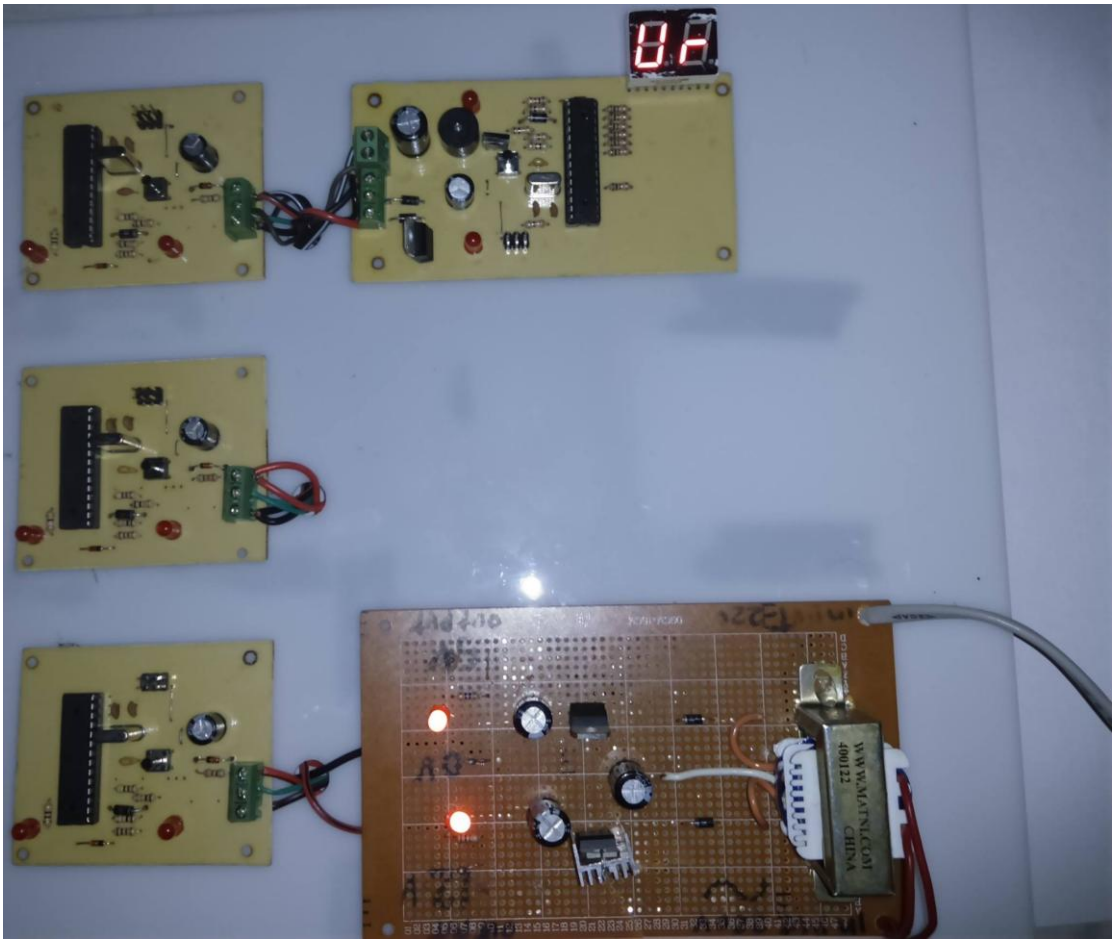


Figure 4.8: Room Urgent Requests a Nurse.

Conclusion and Future Work

The designed system is a multiprocessor distributed system composed of a central device destinated to the nurse room with two digit display, and a number of devices for the patients' rooms. Each of the devices is built around an ATmega8 microcontroller.

A multi-drop bus connects all the system puts derived from the inherent UART in the used microcontroller.

In future we can develop the project to be one of the nurse call system types, such as wireless nurse call system, or we can make a project extension to send an sms or cell phone alarm to notify the nurse where ever they are.

References

[1] M. Verle, "Architecture and programming of 8051 MCU's," 02 05 2015. [Online]. Available: http://www.mikroe.com/.

[2] H. InfoTech, "CodeVisionAVR V3.20," HP, 2000. [Online]. Available: http://www.hpinfotech.ro/. [Accessed 07 05 2015].

[3] CadSoft, "Eagle," CadSoft, 2011. [Online]. Available: http://www.cadsoftusa.com/. [Accessed 07 05 2015].

[4] L. Electronics, "Proteus," Labcenter Electronics, 2009. [Online]. Available: http://www.labcenter.com/index.cfm. [Accessed 07 05 2015].

Appendix A.1 ATmega8 Microcontroller Datasheet

Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 8Kbytes of In-System Self-programmable Flash program memory
  - 512Bytes EEPROM
  - 1Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and QFN/MLF package
    Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    Six Channels 10-bit Accuracy
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V (ATmega8L)
  - 4.5V - 5.5V (ATmega8)
- Speed Grades
  - 0 - 8MHz (ATmega8L)
  - 0 - 16MHz (ATmega8)

Atmel

8-bit Atmel with 8KBytes In-System Programmable Flash

ATmega8
ATmega8L

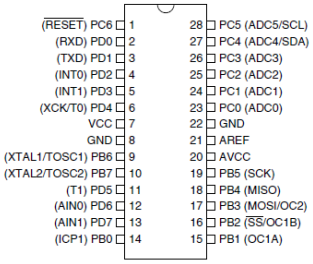
– 0 - 16MHz (ATmega8)
- Power Consumption at 4Mhz, 3V, 25°C
  - Active: 3.6mA
  - Idle Mode: 1.0mA
  - Power-down Mode: 0.5µA

Rev.2486AA–AVR–02/2013

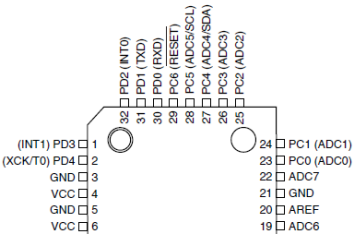
Atmel

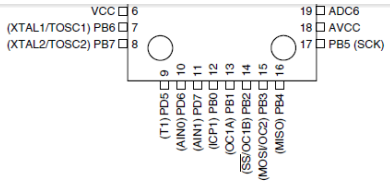
Pin Configurations

PDIP

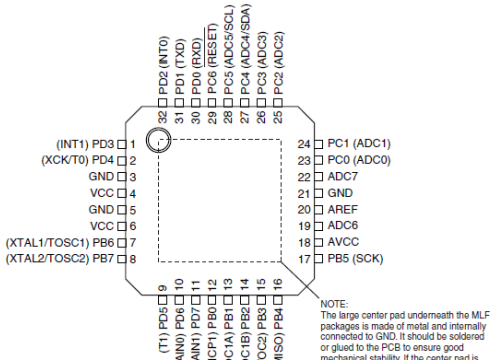


TQFP Top View




VCC 6
(XTAL1/TOSC1) PB6 7
(XTAL2/TOSC2) PB7 8
19 ADC6
18 AVCC
17 PB5 (SCK)
9 (T1) PD5
10 (AIN0) PD6
11 (AIN1) PD7
12 (ICP1) PB0
13 (OC1A) PB1
14 (SS/OC1B) PB2
15 (MOSI/OC2) PB3
16 (MISO) PB4

MLF Top View


32 PD2 (INT0)
31 PD1 (TXD)
30 PD0 (RXD)
29 PC6 (RESET)
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
(INT1) PD3 1
(XCK/T0) PD4 2
GND 3
VCC 4
GND 5
VCC 6
(XTAL1/TOSC1) PB6 7
(XTAL2/TOSC2) PB7 8
24 PC1 (ADC1)
23 PC0 (ADC0)
22 ADC7
21 GND
20 AREF
19 ADC6
18 AVCC
17 PB5 (SCK)
9 (T1) PD5
10 (AIN0) PD6
11 (AIN1) PD7
12 (ICP1) PB0
13 (OC1A) PB1
14 (OC1B) PB2
15 (OC2) PB3
16 (MISO) PB4
NOTE:
The large center pad underneath the MLF
packages is made of metal and internally
connected to GND. It should be soldered
or glued to the PCB to ensure good