IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

DESIGN OF CARDIAC HEALTH MONITORING AND ALERT SYSTEM USING WEARABLE DEVICE

[1]Dr. V. Kumara Swamy, [2]M. Sahithi Thapaswini, [3]A. Manikanta, [4]B. Narasimha
[1]Associate Professor, [2]Student, [3]Student, [4]Student
[1]Electronics and Communication Engineering
[1]Sreenidhi Institute of Science and Technology, Hyderabad, India

ABSTRACT

Technology, which is increasing with time, entirely controls how people live nowadays. It is used all throughout daily life to suit our demands. By utilizing cutting-edge technology, we may enhance both living comfort and measures for health monitoring. In this study, we are using technology to spot serious health conditions so that the patient can get the best treatment possible at the correct moment. While a person is alone at home or on the road, alerting messages about their position are delivered to their mobile phone when their body temperature or heart rate changes. To retrieve the recipient's heart rate, we may also send an SMS using a pre-defined syntax. Here, the GSM modem sends us a warning message, and GPS may be used to determine the person's location. This GPS receiver may provide latitude and longitude information about the place where it was located.

INTRODUCTION

Embedded Systems

An embedded computer system is a system designed to perform one or a small number of targeted tasks, usually with real-time processing constraints. It is a fully functional device, usually incorporating hardware and mechanical components. Personal computers (PCs) are versatile and flexible in design to meet the needs of different users. Some commonly used contemporary products are managed by embedded systems.
An embedded system is controlled by one or more main computing cores, usually a microcontroller or digital signal processor (DSP). Concentration on completing a task (which may require an extremely powerful processor) is an essential quality.

For instance, even though they employ mainframe computers and specialized regional and national networks to connect airports and radar sites, air traffic control systems might be considered embedded. (Each radar most likely has a single or more embedded system.)

Given that the embedded system is dedicated to particular functions, design engineers can optimize it to decrease the product's size and cost while boosting its performance and dependability. Embedded systems are sometimes mass-produced in order to take advantage of economies of scale.

Although most systems have some degree of scalability or programmability, the term "embedded system" cannot be precisely defined. For example, PDAs and embedded systems have operating systems and microprocessors, but PDAs also allow multiple applications to be loaded and peripherals to be connected. Additionally, the system must frequently allow software updates, even if programmability is not exposed as a key feature. Large application systems will often have subcomponents, even if the system as a whole "is designed to perform one or a few specialized functions", making the classification of systems "on a continuum from 'general purpose' to" to "embedded" reasonable to be marked as "embedded".

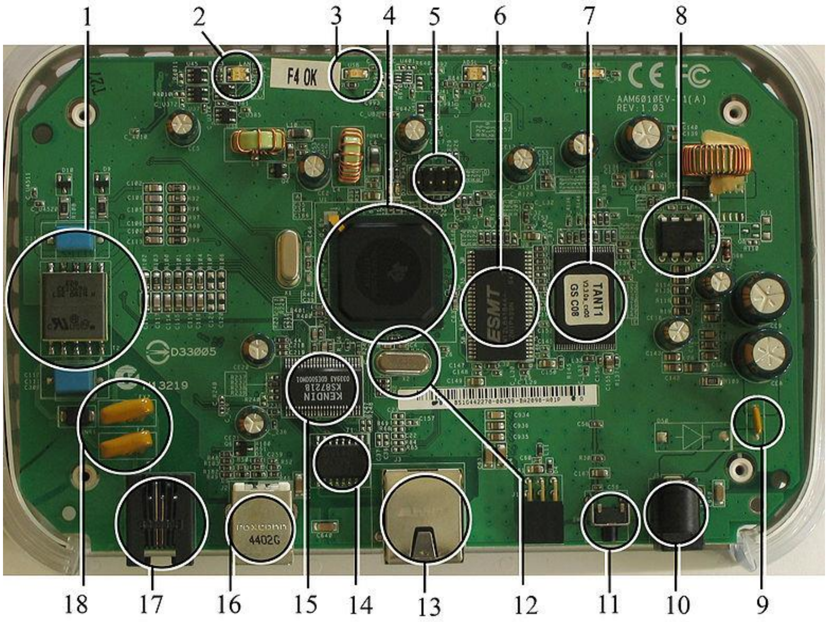


Fig :A modern example of embedded system

Classification of Embedded Systems

Based on performance, functionality, and requirement the embedded systems are divided into three categories:

1. Stand-Alone Embedded System:

These systems receive input in the form of electrical signals from transducers or commands from people, like pressing a button, processing those signals, and then outputting the intended result. Taken together, this process of input processing and output generation operates in standalone mode. These embedded systems are classified as stand-alone embedded systems.
Eg: microwave oven, air conditioner, etc.

2. Real-time embedded systems:

Real-time embedded systems are defined as embedded systems that are utilized to carry out a particular task or activity in a specific amount of time. Real-time embedded systems come in two different flavours.

- Hard Real-time embedded systems:
  These embedded systems operate on an absolute deadline basis, meaning that if the tasking is not completed within a specific time frame, there is a risk of equipment damage.
- Soft Real-Time embedded systems:
  These embedded systems operate under a relative deadline, which means that as long as the task is completed by a certain time, no harm will come to the machine.

3. Network communication embedded systems:

Embedded systems enable a wide range of network-interfaced communication.

Project Overview

In order to do a specific task, an embedded system combines hardware and software. Microcontrollers and microprocessors are two of the most common components found in embedded products. Microprocessors are frequently referred to as general-purpose processors because they only accept inputs, process them, and output the results. A microcontroller, in contrast, not only accepts data as inputs but also manipulates it, interfaces it with different devices, controls it, and ultimately produces the desired outcome.

The PIC16F72 microcontroller is used only in the project "Design of Cardiac Health Monitoring and Alert System using Wearable Device" to keep track of cardiac health and send out alerts through a GSM modem in case of emergencies. The purpose of a GPS receiver is to locate the user on the planet in relation to their position. By utilizing the data it receives from the satellites, the GPS is able to deliver this information. It is a wearable gadget for measuring the heartbeat and body temperature of people all the time.

HARDWARE DESCRIPTION

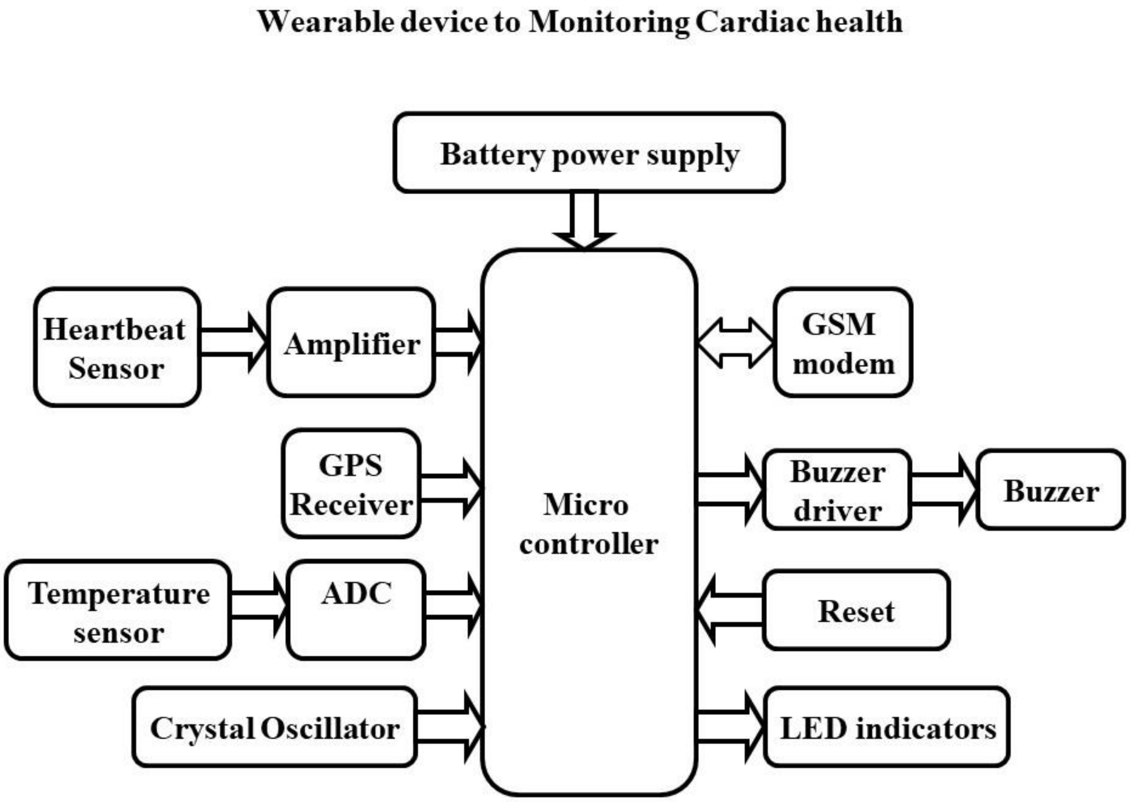


Fig: Block diagram of Wearable device to Monitor Cardiac health

The main blocks of this project are:

1. Microcontroller (16F72)

2. Reset button

3. Crystal oscillator

4. Battery power supply (RPS)

5. LED indicator.

6. GPS module.

7. GSM modem.

8. Heart beat sensor.

9. Buzzer.

10. Temperature sensor.

1. MICROCONTROLLER

Fig: Microcontrollers

A microcontroller is a digital processor with programmable capabilities and the necessary peripherals. Both microcontrollers and microprocessors are very complex sequential digital circuits designed to do the task given by the program or instructions. An analog input/output interface (both analog and digital in nature) can occasionally be found in a microcontroller's mixed-mode circuit.

1. A smaller computer

2. On-chip RAM, ROM, I/O ports...

Example: Motorola’s 6811, Intel’s 8051, Zilog’s Z8, and PIC 16

PIC Microcontrollers:

Peripheral Interface Controller, or PIC, is the designation given by Microchip Technologies to its single-chip microcontrollers. The success of these components in 8-bit microcontrollers is really impressive. The primary factor is that Microchip Technology has consistently improved the device design and equipped the microcontroller with the necessary peripherals to meet customer demands. On the internet, at www.microchip.com, programming tools like assembler and simulator are freely accessible.

Low-end PIC Architectures:

Microchip PIC microcontrollers come in many different varieties. When the PIC microcontroller MCU was initially made accessible by General Instruments in the 1980s, it was a primitive CPU that could only handle 12-bit wide instructions and had just the most fundamental I/O capabilities.
Low-end architectures are these gadgets. They are designed for applications requiring minimal interface functions, little program and data memory, and restricted program memory. Devices with low-end numbers include 12C5XX, 16C5X, and 16C505**.**

Mid-range PIC Architectures

By adding extra peripherals, registers, and data/program memory to low-end PIC architectures, mid-range PIC architectures are created. 16C6X, 16C7X, and 16F87X are a few examples of mid-range devices.

PIC Microcontroller Clock

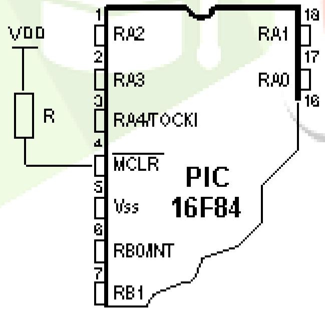
The majority of PIC microcontrollers can run at up to 20MHz. There are four clock cycles in a single instructions cycle (machine cycle). Relationship between the PIC microcontrollers' instruction cycles and clock cycles

Microcontrollers versus Microprocessors

Microprocessors and microcontrollers are very different from one another. Its functioning comes first and is most significant. A microprocessor cannot function without the addition of memory or components for receiving and transmitting data. That essentially means that the computer's microprocessor is its beating heart.
The microcontroller, on the other hand, is intended to be everything in one. Its application requires no additional external parts because it comes with all required peripherals. As a result, we avoid wasting the time and space needed to build gadgets.

2. RESET

To return the microcontroller to a "known" state, use reset. In practice, this means that, in certain bad circumstances, microcontrollers can behave fairly incorrectly. It has to be reset, which would put all registers in their initial positions, in order to continue functioning properly. A reset can also be used to test out a device as an interruption during program execution or to get a microcontroller ready while loading a program, in addition to being utilized when the microcontroller behaves in an unexpected way.



Using the internal reset circuit

Microcontroller PIC16F84 knows several sources of resets:

a) POR (Power-On Reset) is a power-on reset.

b) Bringing logical zero to the MCLR microcontroller's pin is a routine work-day reset.

c) Reset when in SLEEP mode.

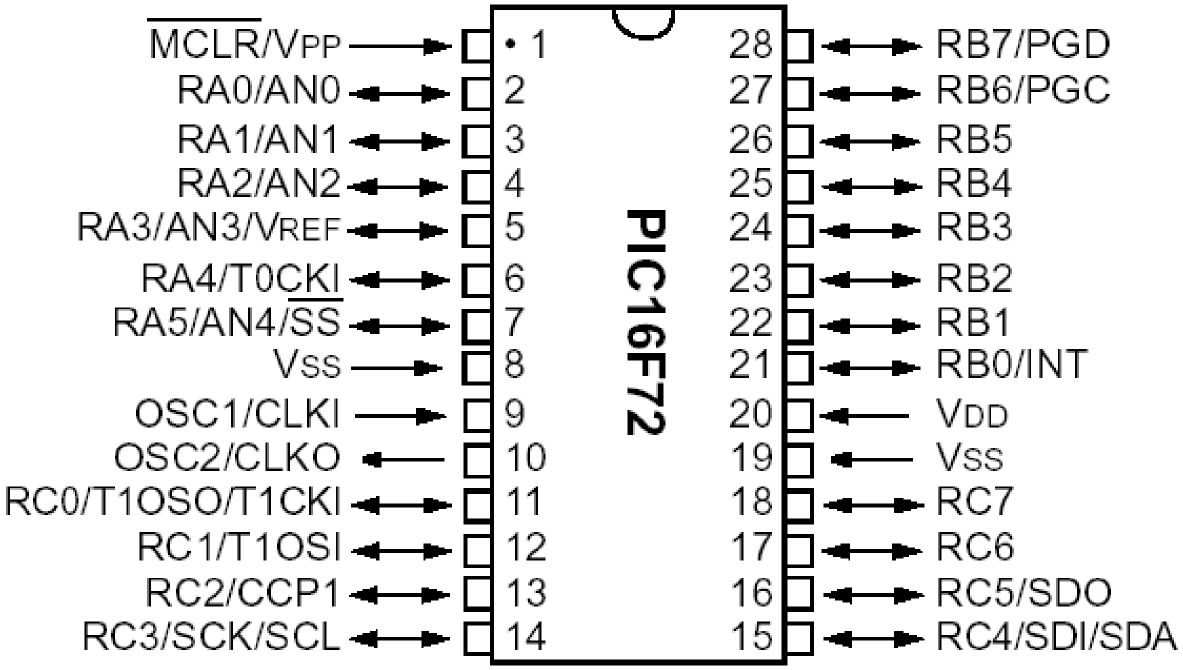
d) Reset at Watchdog Timer Overflow (WDT)

e) Reset during the SLEEP work regime at the WDT overflow.

3. CRYSTAL OSCILLATOR

From DC to 20Mhz, the PIC microcontroller may link to crystal oscillators of various speeds. The 20Mhz oscillator will often be used with the CCS C compiler and is fairly inexpensive. Connecting a 22pF capacitor to the 20 MHz crystal oscillator is recommended. The PIC microcontroller has 5 input/output ports, numbered A through E. Ports B, C, D, and E are the other 4 ports. The role of each port varies. I/O ports are available on the majority of them.

PIN DESCRIPTION



28 pins make up the PIC16F72 in total. The most typical case for it is one of the DIP28 kind, while it can also be found in an SMD case, which is smaller than a DIP. Dual In Package is referred to by the acronym DIP. The acronym SMD stands for Surface Mount Devices, which implies that soldering this kind of component doesn't require mounting holes for pins to pass through.
Pins on PIC16F72 microcontroller have the following meaning:

There are 28 pins on PIC16F72. Most of them can be used as an IO pin. Others are already for specific functions. These are the pin functions.
1. MCLR – to reset the PIC
2. RA0 – port A pin 0
3. RA1 – port A pin 1
4. RA2 – port A pin 2
5. RA3 – port A pin 3
6. RA4 – port A pin 4
7. RA5 – port A pin 5
8. VSS – ground
9. OSC1 – connect to oscillator
10. OSC2 – connect to oscillator
11. RC0 – port C pin 0 VDD – power supply
12. RC1 – port C pin 1
13. RC2 – port C pin 2
14. RC3 – port C pin 3
15. RC4 - port C pin 4
16. RC5 - port C pin 5
17. RC6 - port C pin 6
18. RC7 - port C pin 7
19. VSS - ground
20. VDD – power supply
21. RB0 - port B pin 0
22. RB1 - port B pin 1
23. RB2 - port B pin 2
24. RB3 - port B pin 3
25. RB4 - port B pin 4
26. RB5 - port B pin 5

27. RB6 - port B pin 6
28. RB7 - port B pin 7

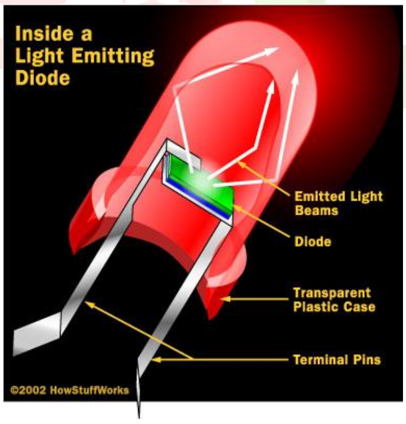
By utilizing all of these pins so many applications can be done such as:
1. LCD – connect to Port B pin.
2. LED – connect to any pin declared as output.
3. Relay and Motor - connect to any pin declared as output.
4. External EEPROM – connect to I2C interface pin – RC3 and RC4 (SCL and SDA)
5. LDR, Potentiometer, and sensor – connect to analog input pins such as RA0.
6. GSM modem dial up modem – connect to RC6 and RC7 – the serial communication interface using RS232 protocol.

4. POWER SUPPLY

3.7V 300mAH (Lithium Polymer) Lipo Rechargeable Battery Model KP-402025

5. LED

A light source made of semiconductors is a light-emitting diode (LED). Several gadgets employ LEDs as indication lamps, and LED lighting is becoming more common. Early LEDs produced dim red light when they were first introduced as a useful electrical component in 1962, but modern LEDs emit bright light that is available in the visible, ultraviolet, and infrared spectrums. The figures depict the internal architecture and component sections of a lead.



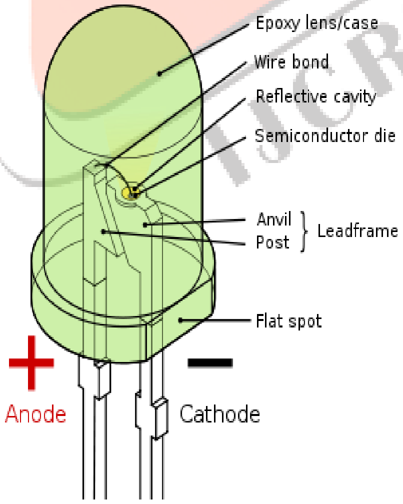


Fig: Inside a LED **Fig: Parts of a LED**

6. GPS MODULE

Global Positioning System

Introduction:

A rapidly developing technology, the Global Positioning System (GPS) offers unparalleled accuracy and flexibility of placement for surveying, GIS data collection, and navigation. A navigation, timing, and positioning system based on satellites is known as GPS NAVSTAR (Navigation Satellite Timing and Ranging Global Positioning System). All across the world, the GPS continuously delivers three-dimensional positioning. In terms of obtaining precise data for navigation up to around 100 meters, mapping at the meter level, and geodetic positioning at the millimeter level, the technology appears to be advantageous to the GPS user group. The collecting of GIS data, surveying, and mapping all benefit greatly from the use of GPS technology.

Satellites in orbit around the Earth, control and observation hubs on Earth, and user-owned GPS receivers are the three components that make up the GPS. GPS receivers detect and identify signals that are broadcast from space by GPS satellites. The time and three-dimensional location (latitude, longitude, and altitude) are then sent by each GPS receiver.

GPS - Components and Basic Facts:

To determine positions anyplace on earth, the GPS uses satellites and computers. Based on satellite ranging, the GPS operates. It follows that the distance from a group of satellites in space is used to calculate the position on the planet. Even while GPS uses some of the most cutting-edge technology ever created, its fundamental concepts are quite simple. The system can be broken down into 5 logical steps to help with basic GPS understanding. Here are some of them:

1. The system's foundation is triangulation from satellites.

2. The GPS calculates the distance based on the radio message's journey time.

3. The GPS requires a very precise clock in order to calculate trip time.

4. After determining a satellite's distance, we must ascertain its location in space.

5. The GPS signal is delayed as it passes through the ionosphere and the earth's atmosphere.

Components of a GPS:

The GPS is divided into three major components

- The Control Segment
- The Space Segments
- The User Segment

The Control Segment:

There are five monitoring stations in the control segment (Colorado Springs, Ascension Island, Diego Garcia, Hawaii, and Kwajalein Island). Ascension, Diego Garcia, and Kwajalein are three of the stations that are used as uplink installations, which can send data to the satellites like new ephemerides (satellite positions as a function of time), clock corrections, and other broadcast message data, while Colorado Springs is used as the master control station.

The Space Segment:

NAVASTAR's constellation of earth-orbiting satellites makes up the space segment. Now, a full constellation of 24 Block II satellites is required by the Defense Department's proposal (21 operational and 3 in-orbit spares). At an inclination of 55 degrees to the equator, the satellites are arranged in 6 orbital planes. They have orbital periods of 12 sidereal hours, or roughly one-half of the earth's periods, or 12 hours of 3-D position fixes. They orbit each other at an altitude of about 12000 miles, at an average distance of about 12000 miles, and at a distance from the stars of about 12000 miles.

The User Segment:

The user segment encompasses the whole civilian and military user and supplier community. All GPS receivers on Earth are included in the User Segment. Receivers vary widely in size and complexity; however, the basic concept is rather simple. A typical receiver consists of an antenna and preamplifier, radio signal processors, a control and display device, a data recording unit, and a power supply. The GPS receiver decodes the timing signals from the 'visible' satellites (four or more) and, after calculating their distances, computes its own latitude, longitude, elevation, and time.

How it works:

GPS satellites circle the world twice a day in a very precise orbit, transmitting signal data to the ground. GPS receivers employ triangulation to calculate the user's precise location based on this information. The GPS receiver, in essence, compares the time a signal was delivered by a satellite to the time it was received. The time difference indicates how far away the satellite is from the GPS receiver. With a few more satellite distance measurements, the receiver can now determine the user's location and show it on the unit's electronic map. To determine a 2D position (latitude and longitude) and track movement, the GPS receiver must lock on to the signal of at least three satellites.

7. GSM

Global System for Mobile Communication (GSM)

Definition:

GSM, which stands for Global System for Mobile Communications, is the most extensively used cell phone technology in the world. Cell phones connect to a cell phone service provider's GSM network by looking for nearby cell phone towers. GSM is a worldwide recognized standard for digital cellular communication.

GSM is the abbreviation for the Global System for Mobile Communications, which was formed in 1982 to develop a uniform European mobile telephone standard that would provide requirements for a pan-European mobile cellular radio system operating at 900 megahertz. More countries outside of Europe are expected to join the GSM alliance.

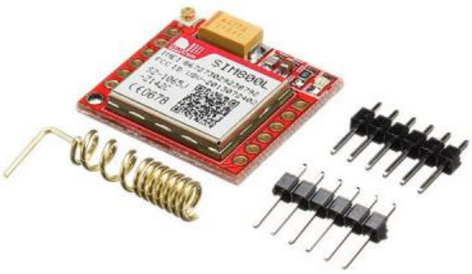


Fig: GSM Module

GSM COMMANDS:

Commands always start with AT (which means Attention) and finish with a <CR> character.

Information responses and result codes

Responses start and end with <CR><LF>, except for the ATV0 DCE response format) and the ATQ1 (result in code suppression) commands.

_ If command syntax is incorrect, an **ERROR** string is returned.

_ If the command syntax is correct but with some incorrect parameters, the **+CME ERROR:**

<Err> or **+CMS ERROR:** <Sms Err> strings are returned with different error codes.

_ If the command line has been performed successfully, an **OK** string is returned.

In some cases, such as “AT+CPIN?” or (unsolicited) incoming events, the product does not return the **OK** string as a response.

In the following examples <CR> and <CR><LF> are intentionally omitted.

1. Manufacturer identification +CGMI

2. Request model identification +CGMM

3. Request revision identification +CGMR

4. Product Serial Number +CGSN

5. Dial command D

ATD<nb> where <nb> is the destination phone number.

6. Hang-Up command H

The ATH (or ATH0) command disconnects the remote user. In the case of multiple calls, all calls are released (active, on hold, and waiting calls).

Syntax:

Command syntax: ATH

7. Answer a call A

When the product receives a call, it sends the ASCII "RING" or "+CRING: type>" text to the application (+CRING if the cellular result code +CRC is enabled). Then it uses the ATA command to wait for the application to accept the call.

Syntax:

Command syntax: ATA

8. Redial last telephone number ATDL

This command dials the number that was last used in the ATD command. For voice calls only, the last number dialed is displayed followed by ";"

Syntax:

Command syntax: ATDL

9. Preferred Message Format +CMGF

Text mode and PDU mode are the message formats that are supported. An entire SMS Message, including all header information, is returned as a binary string in PDU mode (in hexadecimal format). Each pair or letter is transformed into a byte (for example, "41" is changed to the ASCII character "A," which has the ASCII code 0x41 or 65). All commands and responses in Text mode are in ASCII characters. The +CSAS command stores the format selected in EEPROM.

Syntax:

Command syntax: AT+CMGF

10. Read message +CMGR

This command instructs the application to read previously saved messages. The messages are read from the memory that has been specified by the +CPMS command.

Syntax:

Command syntax: AT+CMGR=<index>

11. Send message +CMGS

The <address> parameter contains the address of the terminal to which the message is sent. Simply input the letter ctrl-Z> to send the message (ASCII 26). Except for ctrl-Z> and ESC>, the text can contain all available characters (ASCII 27). When typing text, press the ESC> key to cancel this instruction. In PDU mode, only hexadecimal characters are utilized.

Syntax:

AT+CMGS= <length> <CR>

PDU is entered <ctrl-Z / ESC >

12. Delete message +CMGD

This command deletes one or more messages from the specified message storage ("BM" SMS CB "RAM storage," "SM" SMSPP storage "SIM storage," or "SR" SMS Status-Report storage).

Syntax:

Command syntax: AT+CMGD=<Index> [,<DelFalg>]

8. HEARTBEAT SENSOR

This heart rate sensor is designed to provide a digital output of heat pulses when a finger is placed on it. When the Heart Rate Monitor is active, the top LED indicator flashes smoothly with each heartbeat. This digital output can be connected directly to a microcontroller to measure the beats per minute (BPM) rate. It works by modulating light through blood flow through the finger with each pulse.

Fig: Pulse Sensor

Features

- LED heat beat indicator
- Immediate output digital signal for direct connection to the microcontroller
- Small Dimensions
- Operating Voltage +5V DC

Applications

- Digital Heart Rate Monitor
- Robotics and application biofeedback control
- Workout machines

9. BUZZER

A piezoelectric sound component's sound source is essentially a piezoelectric diaphragm. A piezoelectric diaphragm is made up of two parts: a piezoelectric ceramic plate with electrodes on both sides and a metal plate (brass or stainless steel, etc.). Adhesives hold a piezoelectric ceramic plate to a metal plate. When a direct current voltage is applied between the electrodes of a piezoelectric diaphragm, mechanical distortion occurs due to the piezoelectric effect. In the case of a misshaped piezoelectric element, the distortion expands in a radial direction. The piezoelectric diaphragm bends in the same way. The linked metal plate to the piezoelectric element does not expand. Conversely, when the piezoelectric element contracts, the piezoelectric diaphragm bends in that direction. Thus, when an alternating voltage is applied between the electrodes, bending occurs repeatedly, creating sound waves in the air.

A basic transistor interface circuit is used to interface the buzzer. If the buzzers use different supplies, the 0V rails of each supply must be connected to provide a common reference. If you are using batteries as the power source, remember that piezo sirens draw much less current than buzzers. Sounders also have only one "pitch", whereas piezo sounders can produce sounds of several different pitches.
To switch on the buzzer -high 1
To switch off the buzzer -low 1

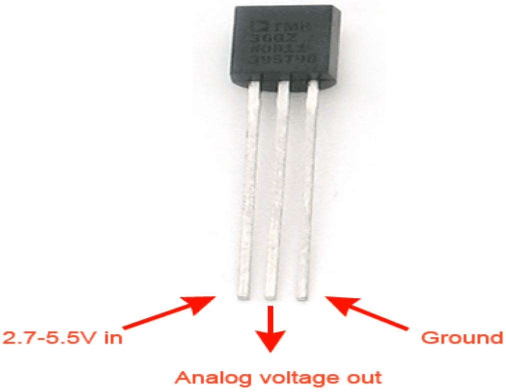
Fig: Buzzer

10. TEMPERATURE SENSOR

Introduction:

LM 35: (TEMPERATURE /FIRE SENSOR):

The LM35 sensor series is a precision integrated-circuit temperature sensor with a voltage output that is proportional to temperature in degrees Celsius. A temperature sensor is used to detect the heat emitted by a fire during sweating.



The LM35 sensor series is a precision integrated-circuit temperature sensor with a directly proportional output voltage to the Celsius temperature (Centigrade).

Description:

In industrialized automation, the temperature is the most commonly measured process variable. A temperature sensor is widely used to convert temperature values to electrical values. Temperature sensors are essential for accurately reading temperatures and controlling temperature in industrial applications.

There is a significant difference in temperature sensor types. Sensors differ greatly in terms of the contact point, temperature range, scaling mechanism, and observing element. Temperature sensors have an observation element housed in plastic or metal housings. The sensor will reflect changes in environmental temperature by altering circuits.

SOFTWARE DESCRIPTION

This project is implemented using the following software:

- Express PCB – for designing circuit
- PIC C compiler - for compilation part

Express PCB

Breadboards are fantastic for prototyping equipment since they allow for tremendous flexibility in modifying a design as needed; however, the final output of a project should preferably have a tidy PCB, few cables, and be able to withstand a shake test. A perfect PCB is not only neater, but it is also more durable because there are no unsecured cables.

Express PCB is a software application used by the firm Express PCB to design PCBs for manufacturing (no other PCB maker accepts Express PCB files). It is simple to use, although it does have certain limits.

- It is more akin to a toy than a professional CAD program.
- It has a limited part library (which we can work around)
- It is unable to import or export files in many formats
- It cannot be used to create DIY production boards.

Several Boards (some multilayer, some surface-mount) have been designed using Express PCB. To manufacture boards, print off PCB layouts and utilize the toner transfer process with an Etch Resistant Pen. Nevertheless, Express PCB lacks a great print arrangement. Here's how to create an Express PCB and tidy up the patterns so they print correctly.

Preparing Express PCB for First Use

The parts list for Express PCB is not that fascinating. Hence, before you begin any project, go to Audio logic and download the additional parts by morsel, ppl, and tangent, then extract them into your Express PCB directory. At this point, launch the software and prepare to configure the workspace to your preferences.

Choose View -> Options. Set the units to "mm" or "in" depending on how you think, and then click "see through the top copper layer" at the bottom. The basic color combination of red and green is commonly utilized, however, it is not as appealing as red and blue.

The Interface

When a project is first started you will be greeted with a yellow outline. This yellow outline is the dimension of the PCB. Typically after positioning of parts and traces, move them to their final position and then crop the PCB to the correct size. However, in designing a board with a certain size constraint, crop the PCB to the correct size before starting.

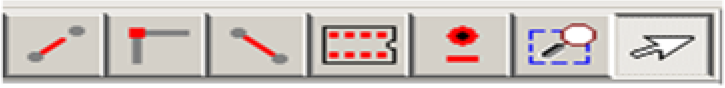


Fig: Tool bar necessary for the interface

Design Considerations

There are numerous approaches to designing a PCB before beginning a project, and one must be picked to meet the needs of the project.

Single-sided, or double-sided?

You can choose to create a single-sided or double-sided PCB when creating one. Single-sided boards are more difficult to design for large projects but are less expensive to produce and easier to etch (e.g. audio signals). A double-sided board is more expensive to build professionally and more challenging to etch on a DIY board, but it simplifies and reduces the size of the component layout.

Ground plane or other special purposes for one side?

You must take into account which traces belong on which side of the board when utilizing a double-sided board. Put power traces on top of the board in general, and vice versa for components that cannot be soldiered onto the top plane (such relays).

PIC Compiler

The machine language code is created and compiled using the PIC compiler program. Following compilation, the machine source code is transformed into hex code that is then loaded into the microcontroller for additional processing. C language code is supported by the PIC compiler as well.

You should be familiar with Embedded C, the C language used by microcontrollers. PIC C is the name we give it because we'll be using the PIC Compiler. Separate compilers exist for the PCB, PCM, and PCH.

For 12-bit, 14-bit, and 16-bit PIC microcontrollers, respectively, PCB, PCM, and PCH are the appropriate opcodes. All three compilers are included in this reference manual because they share many characteristics. These include features and restrictions that only apply to a certain type of microcontroller.

PIC Not much distinguishes C from a typical C program. Writing a C program is not difficult if you are familiar with assembly. We'll have a main function in PIC where you can declare all of your application-specific work.

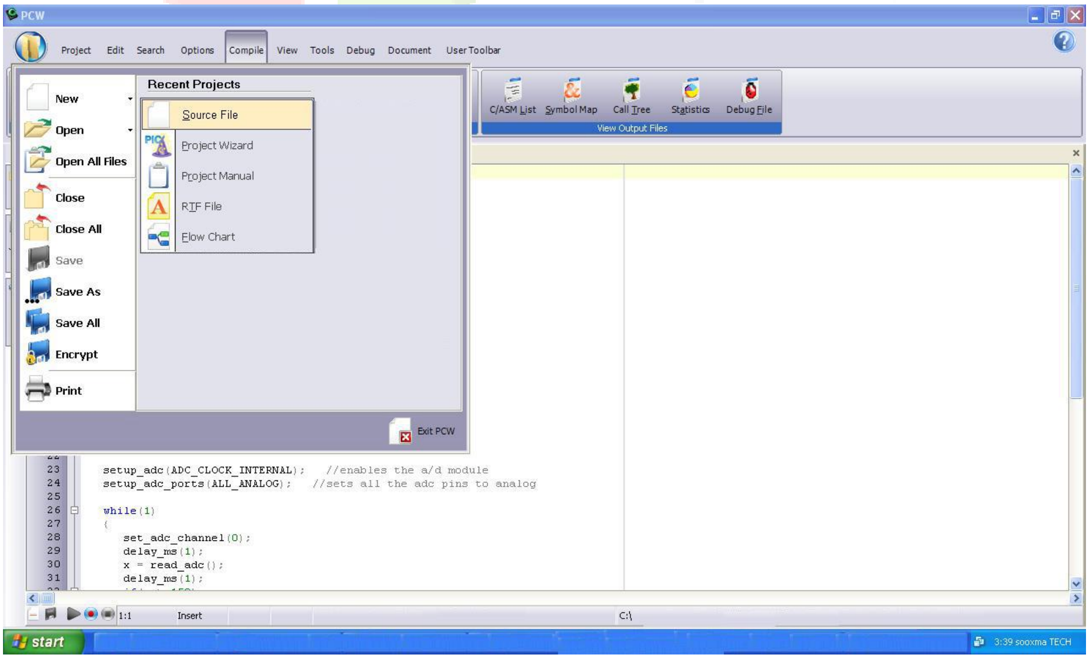
You don't have an operating system running when using embedded C. So, you must ensure that neither your program nor main file ever exits. As they will run indefinitely, a while (1) or for (;;) loop can be used to do this. If a header file isn't added for the controller you're using, you won't be able to access registers for peripherals.

#include <16F877A.h> // header file for PIC 16F877A//

Compilation and simulation steps

The PIC C compiler is used to compile code for the PIC microcontroller. The following are the steps in the compilation:

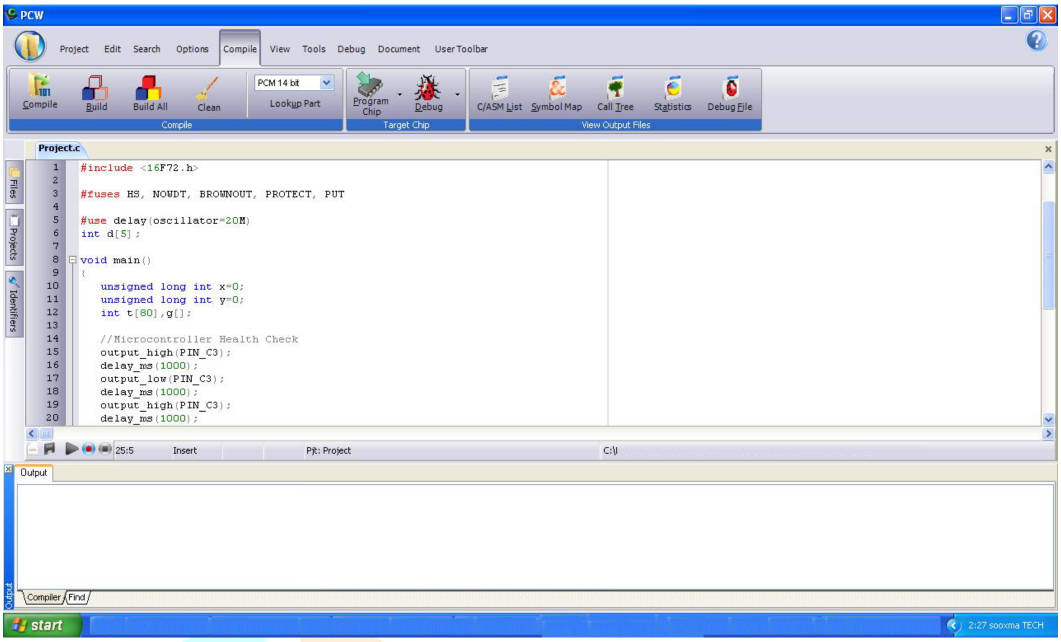
- Open-source PIC C compiler.
- You will be asked to give the new project a name, so create a new folder in which to keep all the project's contents, give it a name, and then click Save.



```
setup_adc(ADC_CLOCK_INTERNAL);   //enables the a/d module
setup_adc_ports(ALL_ANALOG);   //sets all the adc pins to analog

while(1)
{
   set_adc_channel(0);
   delay_ms(1);
   x = read_adc();
   delay_ms(1);
```

Fig: Picture of opening a new file using PIC C compiler

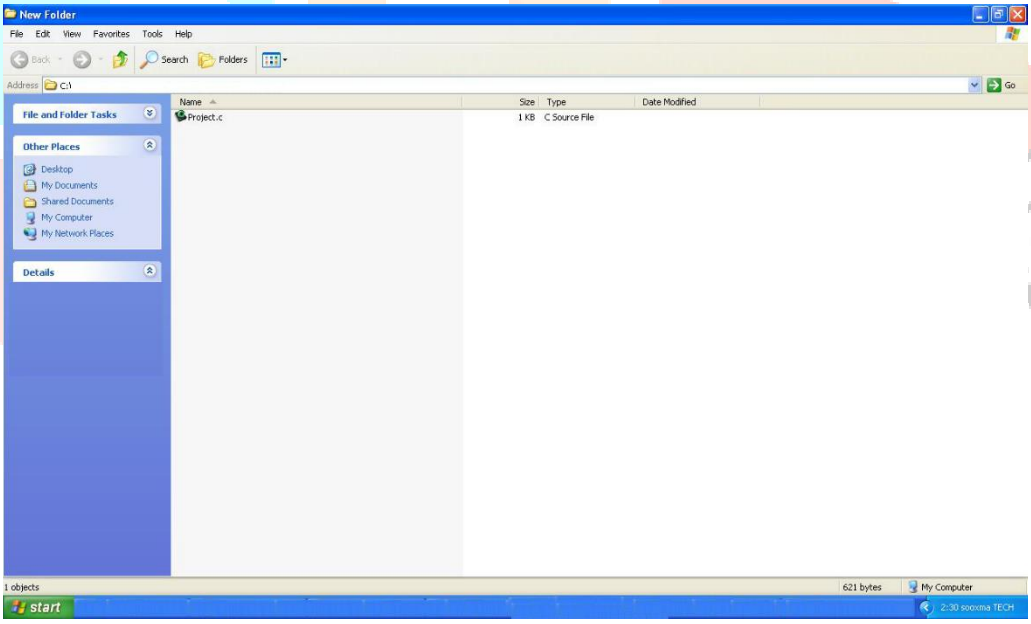
- Choose Project > New, then enter your code in the "Text1" text box that appears.
- Now you have to choose 'File, Save as' and choose a file name for your source code ending with the letter '.c'. Click save after naming the file, "project.c," for example. You must then incorporate this file into your project work.



Project.c

```
#include <16F72.h>

#fuses HS, NOWDT, BROWNOUT, PROTECT, PUT

#use delay(oscillator=20M)
int d[5];

void main()
{
   unsigned long int x=0;
   unsigned long int y=0;
   int t[80],g[];

   //Microcontroller Health Check
   output_high(PIN_C3);
   delay_ms(1000);
   output_low(PIN_C3);
   delay_ms(1000);
   output_high(PIN_C3);
   delay_ms(1000);
```

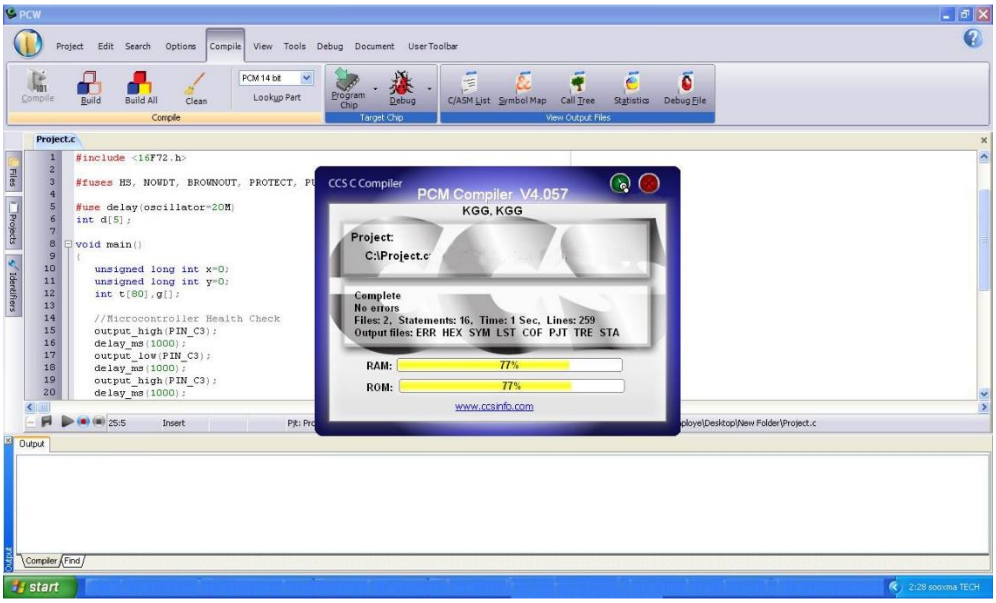
Fig: Picture of compiling a new file using PIC C compiler



Name	Size	Type	Date Modified
Project.c	1 KB	C Source File	

Fig: Picture of compiling a project.c file using PIC C compiler

- You can then begin writing the source code in the window titled "project.c," but before you can test it, you must compile it and fix any potential syntax mistakes.


PCW
Project Edit Search Options Compile View Tools Debug Document User Toolbar
Compile Build Build All Clean PCM 14 bit Lookup Part
Compile
Program Chip Debug
Target Chip
C/ASM List Symbol Map Call Tree Statistics Debug File
View Output Files
Project.c
Files Projects Identifiers
25:5 Insert Pjt: Pr
ploye\Desktop\New Folder\Project.c
Output
Compiler Find
start
2:28 sooxma TECH

```
#include <16F72.h>

#fuses HS, NOWDT, BROWNOUT, PROTECT, PU

#use delay(oscillator=20M)
int d[5];

void main()
{
   unsigned long int x=0;
   unsigned long int y=0;
   int t[80],g[];

   //Microcontroller Health Check
   output_high(PIN_C3);
   delay_ms(1000);
   output_low(PIN_C3);
   delay_ms(1000);
   output_high(PIN_C3);
   delay_ms(1000);
```

CCS C Compiler

PCM Compiler V4.057

KGG, KGG

Project:
C:\Project.c

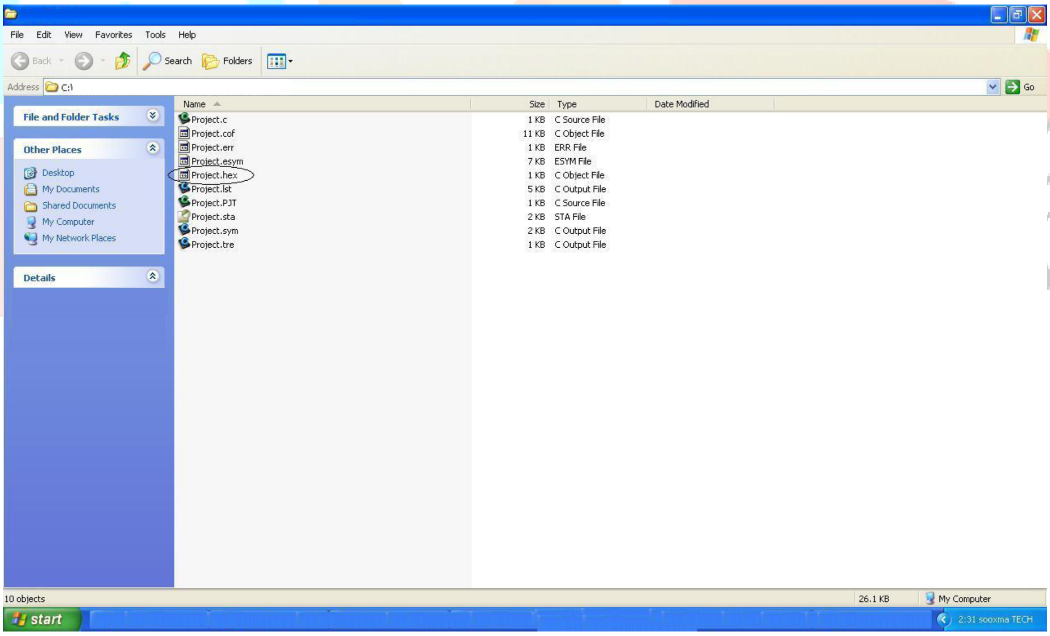
Complete
No errors
Files: 2, Statements: 16, Time: 1 Sec, Lines: 259
Output files: ERR HEX SYM LST COF PJT TRE STA

RAM: 77%
ROM: 77%

www.ccsinfo.com

Fig : Picture of checking errors and warnings using PIC C compiler

- By selecting the option to compile. Hex files are automatically created.
- This is how we compile a program to check for errors, and the generated program is then saved in the file from where the program was originally started.



Name	Size	Type	Date Modified
Project.c	1 KB	C Source File	
Project.cof	11 KB	C Object File	
Project.err	1 KB	ERR File	
Project.esym	7 KB	ESYM File	
Project.hex	1 KB	C Object File	
Project.lst	5 KB	C Output File	
Project.PJT	1 KB	C Source File	
Project.sta	2 KB	STA File	
Project.sym	2 KB	C Output File	
Project.tre	1 KB	C Output File	

Fig: Picture of .hex file existing using PIC C compiler

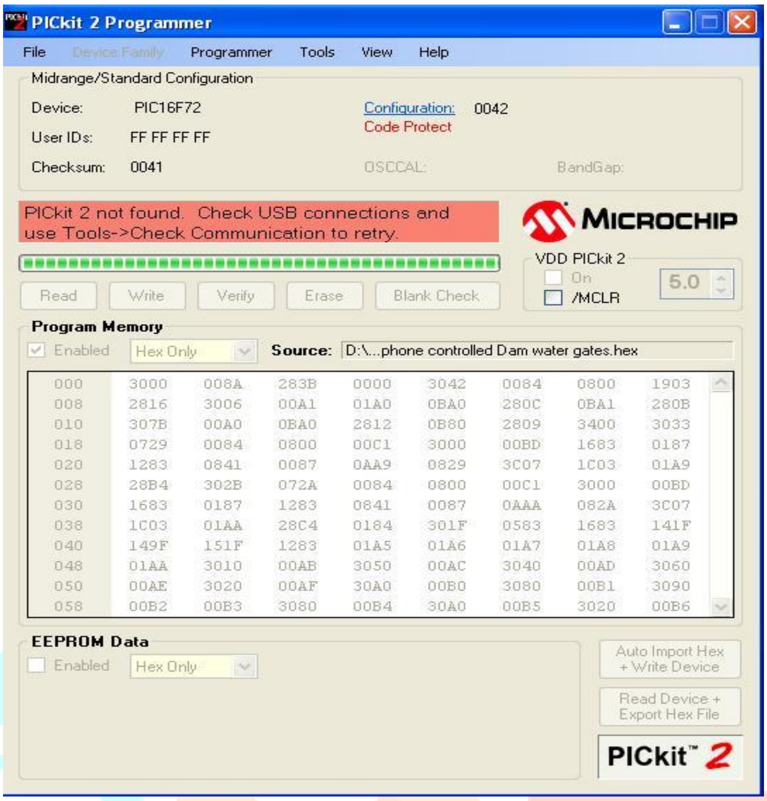
Simulating comes next after compilation. Here, the Proteus 7 software is used to construct the first circuit, which is then followed by simulation and dumping. The steps in the simulation are as follows:

- Click IS1S6 in Proteus 7 after opening it.
- It now shows the PCB on which the microcontroller-designed circuit was built. Circuit design requires the use of components. So, select the component choice.
- Next, choose PIC16F877A from the list of project-related components under the letter "p," and then click OK. Your "Target device," which is the PIC 16F877A, is where your source code will ultimately run.

Dumping steps:

The following procedures are necessary to dump the program modified in Proteus 7 to a microcontroller:

1. The popup was initially displayed prior to connecting the program dumper to the microcontroller kit as seen below.



Midrange/Standard Configuration

Device: PIC16F72

User IDs: FF FF FF FF

Checksum: 0041

Configuration: 0042

Code Protect

OSCCAL:

BandGap:

PICkit 2 not found. Check USB connections and use Tools->Check Communication to retry.

VDD PICkit 2

□ On

□ /MCLR

5.0

Program Memory

☑ Enabled Hex Only

Source: D:\...phone controlled Dam water gates.hex

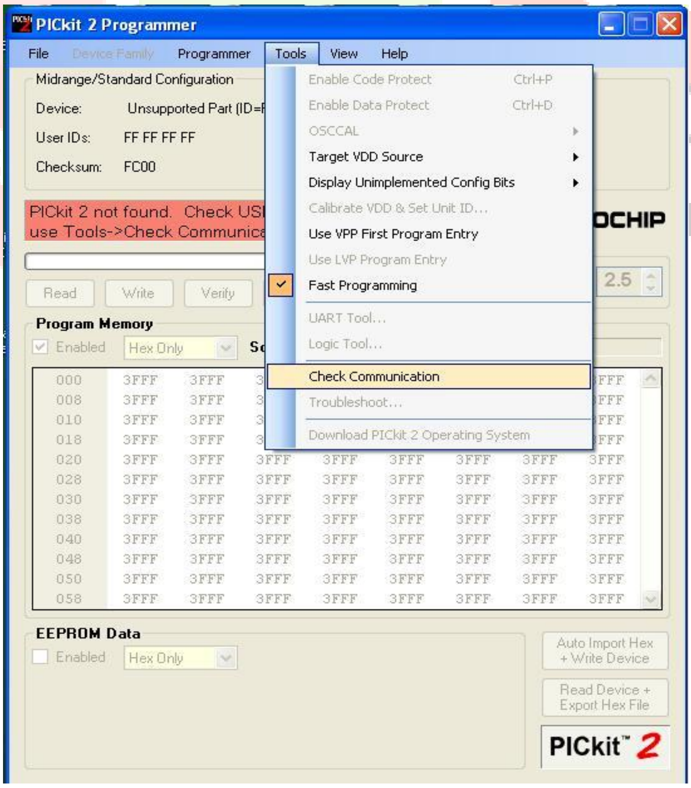
```
000   3000  008A  283B  0000  3042  0084  0800  1903
008   2816  3006  00A1  01A0  0BA0  280C  0BA1  280B
010   307B  00A0  0BA0  2812  0B80  2809  3400  3033
018   0729  0084  0800  00C1  3000  00BD  1683  0187
020   1283  0841  0087  0AA9  0829  3C07  1C03  01A9
028   28B4  302B  072A  0084  0800  00C1  3000  00BD
030   1683  0187  1283  0841  0087  0AAA  082A  3C07
038   1C03  01AA  28C4  0184  301F  0583  1683  141F
040   149F  151F  1283  01A5  01A6  01A7  01A8  01A9
048   01AA  3010  00AB  3050  00AC  3040  00AD  3060
050   00AE  3020  00AF  30A0  00B0  3080  00B1  3090
058   00B2  00B3  3080  00B4  30A0  00B5  3020  00B6
```

EEPROM Data

□ Enabled Hex Only

Fig: Picture of program dumper window

2. To establish a connection, choose the Tools option and click Check Communication as displayed in the window below.


PICkit 2 Programmer
File
Device Family
Programmer
Tools
View
Help
Enable Code Protect Ctrl+P
Enable Data Protect Ctrl+D
OSCCAL
Target VDD Source
Display Unimplemented Config Bits
Calibrate VDD & Set Unit ID...
Use VPP First Program Entry
Use LVP Program Entry
✓ Fast Programming
UART Tool...
Logic Tool...
Check Communication
Troubleshoot...
Download PICkit 2 Operating System
OCHIP
Read
Write
Verify
Auto Import Hex + Write Device
Read Device + Export Hex File
PICkit™ 2

Midrange/Standard Configuration

Device: Unsupported Part (ID=F

User IDs: FF FF FF FF

Checksum: FC00

PICkit 2 not found. Check USI use Tools->Check Communica

2.5

Program Memory

☑ Enabled Hex Only

```
000   3FFF  3FFF  3                                 FFF
008   3FFF  3FFF  3                                 FFF
010   3FFF  3FFF  3                                 FFF
018   3FFF  3FFF  3                                 FFF
020   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
028   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
030   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
038   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
040   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
048   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
050   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
058   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
```

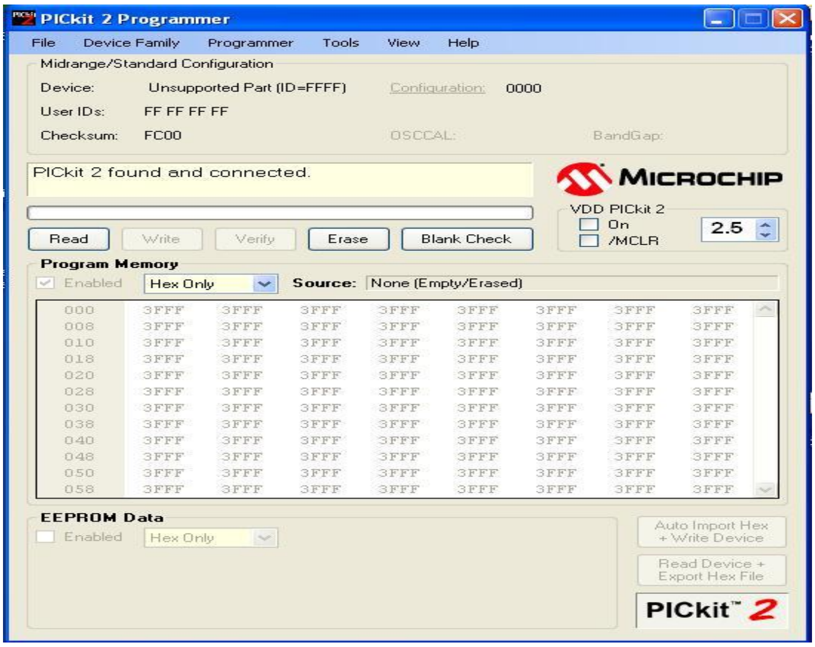
EEPROM Data

□ Enabled Hex Only

Fig : Picture of checking communications before dumping program into microcontroller

3. The window that appears after successfully connecting the dumper to the microcontroller kit is shown below.

Fig: Picture after connecting the dumper to microcontroller



Device: Unsupported Part (ID=FFFF) Configuration: 0000

User IDs: FF FF FF FF

Checksum: FC00 OSCCAL: BandGap:

PICkit 2 found and connected.

VDD PICkit 2 □ On □ /MCLR 2.5

Program Memory

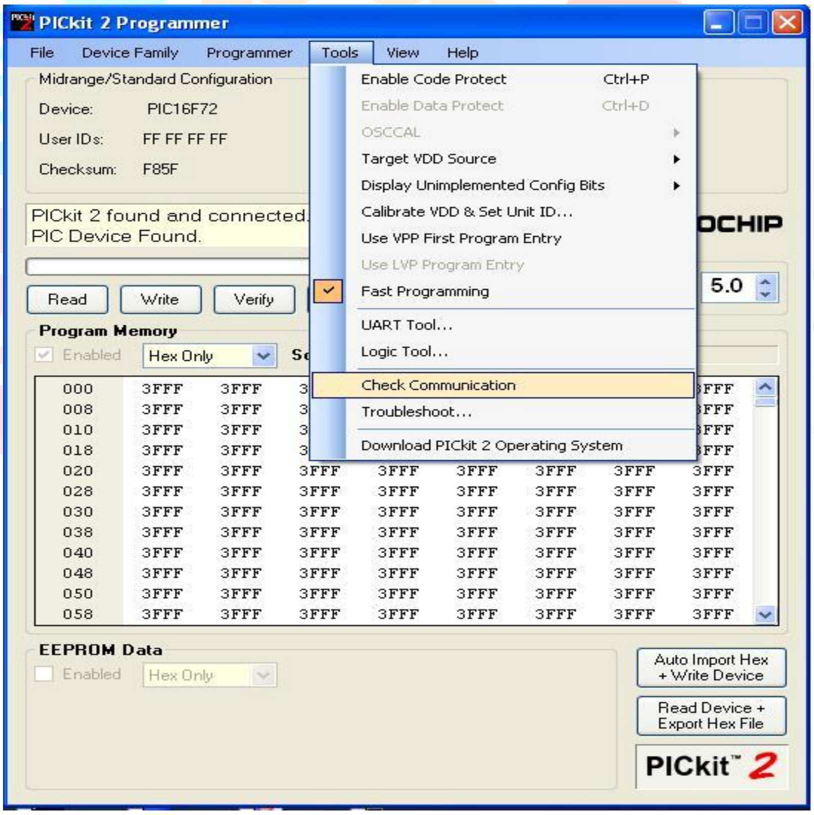
☑ Enabled Hex Only Source: None (Empty/Erased)

```
000   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
008   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
010   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
018   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
020   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
028   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
030   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
038   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
040   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
048   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
050   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
058   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
```

EEPROM Data

□ Enabled Hex Only

4. The microcontroller is again identified by the dumper by selecting the Tools option and clicking on Verify Communication; as a result, the window is as shown below.


PICkit 2 Programmer
File
Device Family
Programmer
Tools
View
Help
Midrange/Standard Configuration
Enable Code Protect Ctrl+P
Enable Data Protect Ctrl+D
OSCCAL
Target VDD Source
Display Unimplemented Config Bits
Calibrate VDD & Set Unit ID...
Use VPP First Program Entry
Use LVP Program Entry
Fast Programming
UART Tool...
Logic Tool...
Check Communication
Troubleshoot...
Download PICkit 2 Operating System
OCHIP
Read
Write
Verify
Auto Import Hex + Write Device
Read Device + Export Hex File
PICkit™ 2

Device: PIC16F72

User IDs: FF FF FF FF

Checksum: F85F

PICkit 2 found and connected.
PIC Device Found.

5.0

Program Memory

☑ Enabled Hex Only So

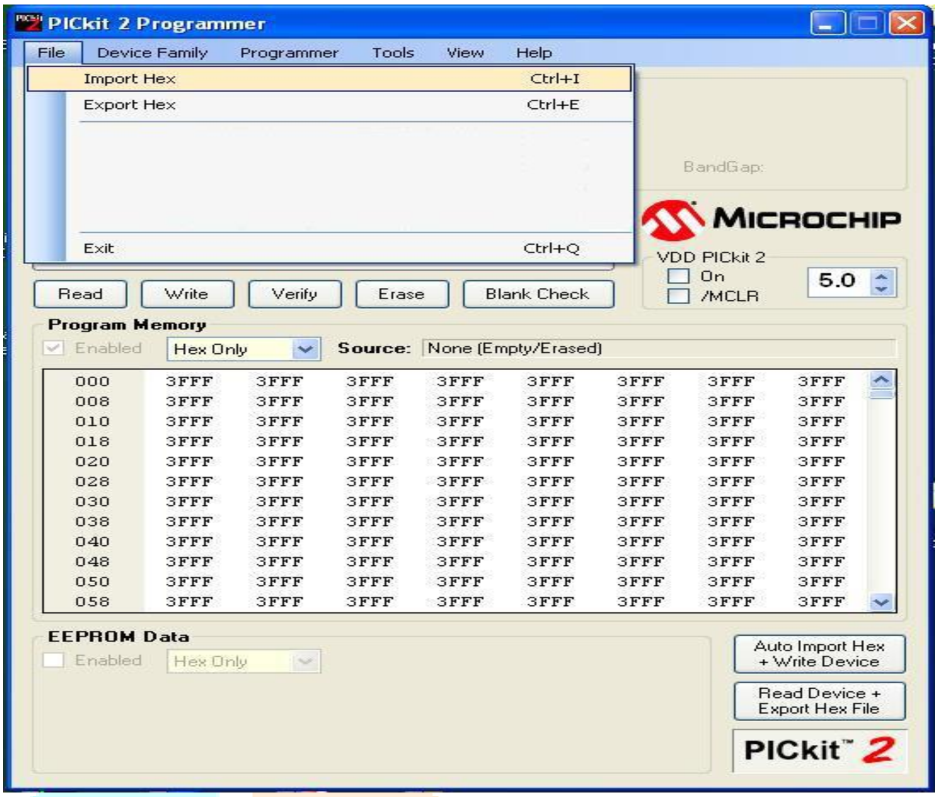
```
000   3FFF   3FFF   3                                          FFF
008   3FFF   3FFF   3                                          FFF
010   3FFF   3FFF   3                                          FFF
018   3FFF   3FFF   3                                          FFF
020   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
028   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
030   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
038   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
040   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
048   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
050   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
058   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF   3FFF
```

EEPROM Data

□ Enabled Hex Only

Fig: Picture of dumper recognition to the microcontroller

5. Import the program, which is a ".hex" file, from the saved location by choosing File > Import > Hex as displayed in the window below.



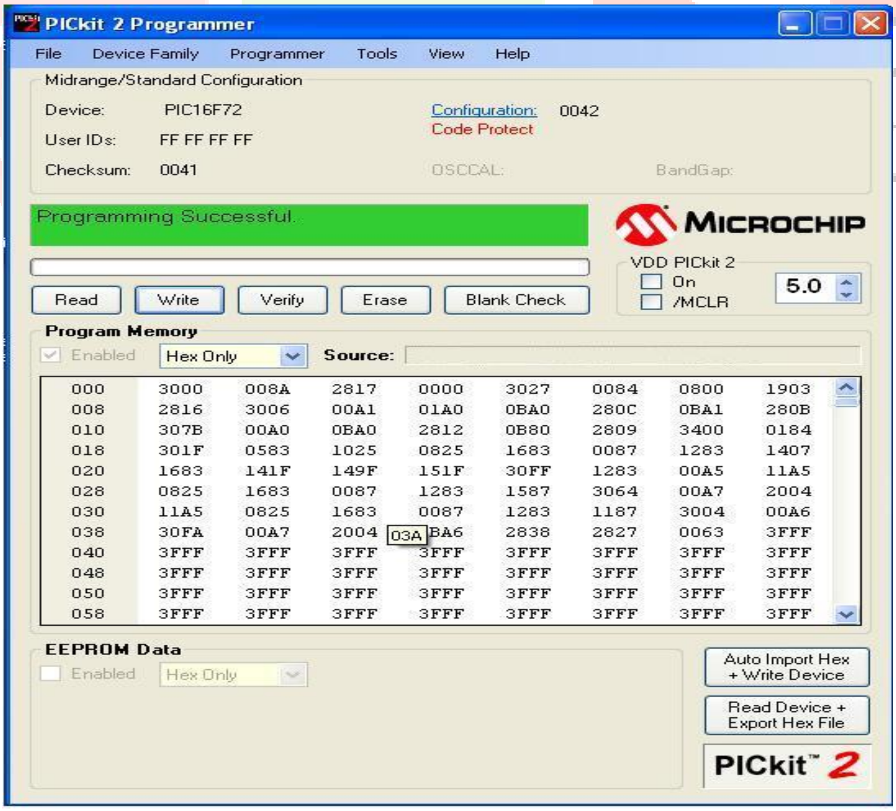
Source: None (Empty/Erased)

```
000   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
008   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
010   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
018   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
020   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
028   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
030   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
038   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
040   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
048   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
050   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
058   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
```

Fig: Picture of program importing into the microcontroller

6. After selecting the "Import Hex" option, we must navigate to the program's location, select "prog.hex," and then click "open" to upload the program to the microcontroller.

7. After a successful program dump, the window looks like the one below.



Device: PIC16F72

User IDs: FF FF FF FF

Checksum: 0041

Configuration: 0042

Code Protect

OSCCAL:

BandGap:

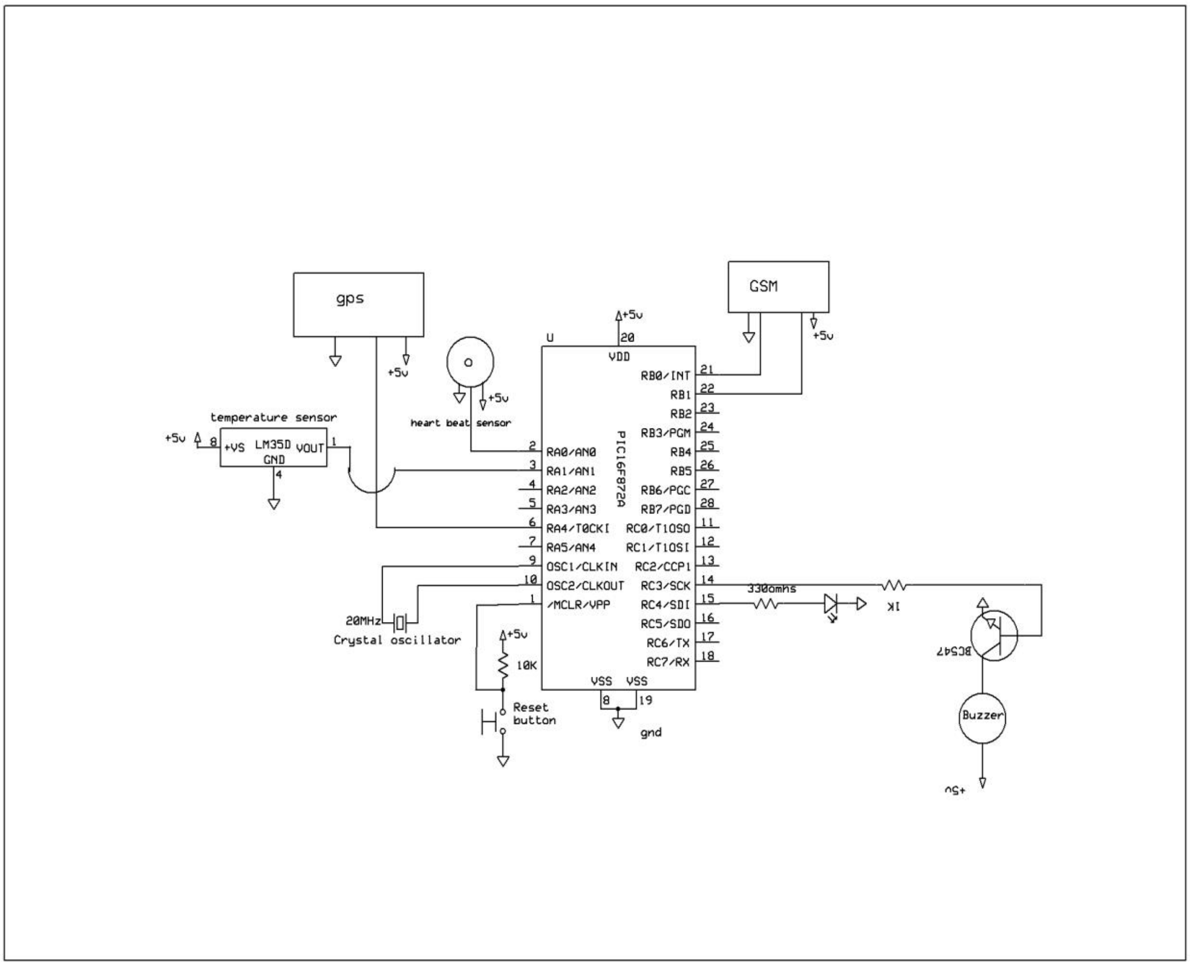
Programming Successful.

Source:

```
000   3000  008A  2817  0000  3027  0084  0800  1903
008   2816  3006  00A1  01A0  0BA0  280C  0BA1  280B
010   307B  00A0  0BA0  2812  0B80  2809  3400  0184
018   301F  0583  1025  0825  1683  0087  1283  1407
020   1683  141F  149F  151F  30FF  1283  00A5  11A5
028   0825  1683  0087  1283  1587  3064  00A7  2004
030   11A5  0825  1683  0087  1283  1187  3004  00A6
038   30FA  00A7  2004 03A BA6  2838  2827  0063  3FFF
040   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
048   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
050   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
058   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
```

Fig: Picture after program dumped into the microcontroller

PROJECT DESCRIPTION



In this, the schematic diagram and interfacing of the PIC16F72 microcontroller with each module are considered.

Fig: schematic diagram of Wearable device to Monitor Cardiac health

The schematic illustration of the top Wearable gadget to monitor cardiac health illustrates how each component is interfaced with a microcontroller and GPS module to identify the position according to the user's needs. A crystal oscillator is attached to the microcontroller's pins 9 and 10.

ADVANTAGES

- Design a medical monitoring system that can be worn.
- SMS and buzzer alerts for unusual incidents.
- Mechanism for observing position and health continuously.
- A fair price.

DISADVANTAGES

- A properly installed heartbeat sensor will function properly.
- The network signal affects how the GSM module functions.

APPLICATIONS

- Real-time human health monitoring.
- Child health monitoring.

RESULT

The goal of the "Design of Cardiac Health Monitoring and Alert System using Wearable Device" project was to create a wearable device that could be used to find cardiac patients using a GPS module. The wearable device also tracks the patient's heart rate, alerts in an emergency with a buzzer alarm system, and uses a GSM modem to send SMS to predetermined contacts.

CONCLUSION

It is created by integrating features for each hardware element used. The existence of each module has been carefully studied and organized in order to obtain the best performance from the equipment.

Second, the project has been successfully implemented using the latest integrated circuits and ever-evolving technology. Therefore, the project was efficiently designed and tested.

REFERENCES

1. Bayoumy, K.; Gaber, M.; Elshafeey, A.; Mhaimeed, O.; Dineen, E.H.; Marvel, F.A.; Martin, S.S.; Muse, E.D.; Turakhia, M.P.; Tarakji, K.G.; et al. Smart Wearable Devices in Cardiovascular Care: Where We Are and How to Move Forward. Nat. Rev. Cardiol. 2021, 18, 581–599. [CrossRef]

2. Soon, S.; Svavarsdottir, H.; Downey, C.; Jayne, D.G. Wearable Devices for Remote Vital Signs Monitoring in the Outpatient Setting: An Overview of the Field. BMJ Innov. 2020, 6, 55–71. [CrossRef]

3. Gordan, R.; Gwathmey, J.K.; Xie, L.H. Autonomic and Endocrine Control of Cardiovascular Function. World J. Cardiol. 2015, 7, 204–214. [CrossRef] [PubMed]

4. Zhang, D.; Wang, W.; Li, F. Association between Resting Heart Rate and Coronary Artery Disease, Stroke, Sudden Death and Noncardiovascular Diseases: A Meta-Analysis. CMAJ 2016, 188, E384–E392. [CrossRef] [PubMed]

5. Singh, N.; Moneghetti, K.J.; Christle, J.W.; Hadley, D.; Plews, D.; Froelicher, V. Heart Rate Variability: An Old Metric with New Meaning in the Era of Using MHealth Technologies for Health and Exercise Training Guidance. Part One: Physiology and Methods. Arrhythm. Electrophysiol. Rev. 2018, 7, 193–198. [CrossRef] [PubMed]

6. Fox, K.; Ford, I.; Steg, P.G.; Tendera, M.; Robertson, M.; Ferrari, R.; BEAUTIFUL Investigators. Heart Rate as a Prognostic Risk Factor in Patients with Coronary Artery Disease and Left-Ventricular Systolic Dysfunction (BEAUTIFUL): A Subgroup Analysis of a Randomised Controlled Trial. Lancet 2008, 372, 817–821. [CrossRef]

7. Attia, Z.I.; Noseworthy, P.A.; Lopez-Jimenez, F.; Asirvatham, S.J.; Deshmukh, A.J.; Gersh, B.J.; Carter, R.E.; Yao, X.; Rabinstein, A.A.; Erickson, B.J.; et al. An Artificial Intelligence-Enabled ECG Algorithm for the Identification of Patients with Atrial Fibrillation during Sinus Rhythm: A Retrospective Analysis of Outcome Prediction. Lancet 2019, 394, 861–867. [CrossRef]

8. Yan, B.P.; Lai, W.H.S.; Chan, C.K.Y.; Chan, S.C.; Chan, L.; Lam, K.; Lau, H.; Ng, C.; Tai, L.; Yip, K.; et al. Contact-Free Screening of Atrial Fibrillation by a Smartphone Using Facial Pulsatile Photoplethysmographic Signals. J. Am. Heart. Assoc. 2018, 7, e008585. [CrossRef] [PubMed]

9. Shaffer, F.; Ginsberg, J.P. An Overview of Heart Rate Variability Metrics and Norms. Front. Public Health 2017, 258. [CrossRef]

10. Task Force of the European Society of Cardiology the North American Society of Pacing Electrophysiology Task Force of the European Society of Cardiology and the North American Society of Pacing And. Circulation 1996, 93, 1043–1065.

11. Laborde, S.; Mosley, E.; Thayer, J.F. Heart Rate Variability and Cardiac Vagal Tone in Psychophysiological ResearchRecommendations for Experiment Planning, Data Analysis, and Data Reporting. Front. Psychol. 2017, 8, 213. [CrossRef] [PubMed]

12. Bauer, A.; Kantelhardt, J.W.; Barthel, P.; Schneider, R.; Mäkikallio, T.; Ulm, K.; Hnatkova, K.; Schömig, A.; Huikuri, H.; Bunde, A.; et al. Deceleration Capacity of Heart Rate as a Predictor of Mortality after Myocardial Infarction: Cohort Study. Lancet 2006, 367, 1674–1681. [CrossRef]

13. Schmidt, G.; Malik, M.; Barthel, P.; Schneider, R.; Ulm, K.; Rolnitzky, L.; Camm, A.J.; JT Jr, B.; Schömig, A. Heart-Rate Turbulence after Ventricular Premature Beats as a Predictor of Mortality after Acute Myocardial Infarction. Lancet 1999, 353, 1390–1396. [CrossRef]

14. Grimm, W.; Schmidt, G.; Maisch, B.; Sharkova, J.; Müller, H.H.; Christ, M. Prognostic Significance of Heart Rate Turbulence Following Ventricular Premature Beats in Patients with Idiopathic Dilated Cardiomyopathy. J. Cardiovasc. Electrophysiol. 2003, 14, 819–824.