Real – Time Wireless Embedded Electronics For Solider Security

G. AKSHITH REDDY
B.TECH
JBIET
P. NARESH REDDY
B.TECH
JBIET
R. PRASHANTH KUMAR
B.TECH
JBIET

ABSTRACT

Various therapeutic applications set new demands on sensor sort out styles. They routinely incorporate incredibly factor information rates, various recipients and security. Most existing discoverer compose styles don't adequately reinforce these necessities, focusing rather on collecting little proportions of data from centre points while not security. During this paper, we will when all is said in done give a pack style for restorative marker frame works. A crisis facility parental figure will approach this| data at any explanation in time and shouldn't be obliged to be physically present inside the patient. The framework establishment canters district unit self-powered and draw essentialness from overhead 34W glaring lights by methods for sun arranged sheets. The device centre points are oftentimes interfaced to a spread of immense sign sensors like electrocardiograms (ECGs), Heart beat meters and weight level (BP) sensors. So as to check a totally working system, a business BP/beat screen (BPM) was interfaced to a remote contraption node. To start an examining, by then assembles the information and advances it to the base station. A charming graphical PC program (GUI) was expected to store and show understanding data on the base station laptop. The set-up was viewed as exceptionally strong with low power use .This structure gives a social affair of shows and organizations unequivocally custom fitted for this application space. It consolidates a protected correspondences model, AN interface for incidental gathering of locator information, a ground-breaking discoverer exposure show and shows that screen and expel to seventieth of the structure is made in Tiny OS and a JAVA chiefly based program is given to address the framework and show the intentional information. Likewise, all around examination of the structure of a 6-center locator demonstrating ground is introduced, estimation quantifiability and strength considering the way that the extent of sensors and thusly the per centre point rate. The results show that the foreseen structure could be flexible, groundbreaking, strong and

secure objectives for remedial applications.

INTRODUCTION

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. A good example is the microwave oven. Almost every household has one, and tens of millions of them are used every day, but very few people realize that a processor and software are involved in the preparation of their lunch or dinner.

This is in direct contrast to the personal computer in the family room. It too is comprised of computer hardware and software and mechanical components (disk drives, for example). However, a personal computer is not designed to perform a specific function rather; it is able to do many different things. Many people use the term general-purpose computer to make this distinction clear. As shipped, a general-purpose computer is a blank slate; the manufacturer does not know what the customer will do wish it. One customer may use it for a network file server another may use it exclusively for playing games, and a third may use it to write the next great American novel.

Frequently, an embedded system is a component within some larger system. For example, modern cars and trucks contain many embedded systems. One embedded system controls the anti-lock brakes, other monitors and controls the vehicle's emissions, and a third displays information on the dashboard. In some cases, these embedded systems are connected by some sort of a communication network, but that is certainly not a requirement.

At the possible risk of confusing you, it is important to point out that a general-purpose computer is itself made up of numerous embedded systems. For example, my computer consists of a keyboard, mouse, video card, modem, hard drive, floppy drive, and sound card- each of which is an embedded system. Each of these devices contains a processor and software and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over analog telephone line. That's it and all of the other devices can be summarized in a single sentence as well.

If an embedded system is designed well, the existence of the processor and software could be completely unnoticed by the user of the device. Such is the case for a microwave oven, VCR, or alarm clock. In some cases, it would even be possible to build an equivalent device that does not contain the processor and software. This could be done by replacing the combination with a custom integrated circuit that performs the same functions in hardware.

What are microcontrollers and what are they used for?

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.

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 11 illiseconds 11 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" appliences.
- 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" raedy-to-use programs into the microcontroller.

Where is real time wireless embedded electronics for soldier security is used?

Real-time wireless embedded electronics for soldier security is used in various military and defense applications to enhance situational awareness, communication, and safety. Some key areas where it is implemented include:

1. **Battlefield Monitoring –** Wearable sensors track soldiers' vital signs (heart rate, temperature, hydration) and environmental conditions in real time.
2. **GPS and Location Tracking** – Embedded GPS systems help commanders monitor troop movements and provide navigation assistance.
3. **Wearable Communication Systems** – Advanced wireless communication devices

allow secure and real-time voice and data transmission.

4. **Smart Armor and Clothing –** Embedded sensors in body armor detect bullet impacts and injuries, sending alerts to medics.
5. **Drone and UAV Integration –** Soldiers use wearable devices to communicate with drones for reconnaissance and surveillance.
6. **Threat Detection and Warning Systems** – Sensors detect nearby threats like chemical, biological, or radioactive hazards and provide alerts.
7. **Augmented Reality (AR) Headsets –** Used for real-time battlefield visualization, target identification, and mission planning.
8. **Firearm and Equipment Tracking** – Wireless sensors monitor the status of weapons and gear for efficiency and security.

Common Applications of Real time wireless embedded electronics for soldier security

1. **Real-Time Health Monitoring –** Wearable sensors track vital signs (heart rate, temperature, hydration, stress levels) and detect injuries or fatigue.
2. **GPS and Location Tracking** – Embedded GPS and inertial navigation systems provide real-time location tracking for soldiers, improving coordination and rescue operations.
3. **Communication Systems** – Secure wireless networks enable real-time communication between soldiers and command centers, ensuring efficient coordination and situational awareness.
4. **Smart Body Armor –** Sensors embedded in body armor detect impacts, assess injury severity, and alert medics in real time.

BLOCK DIAGRAM AND ITS WORKING

BLOCKDIAGRAM

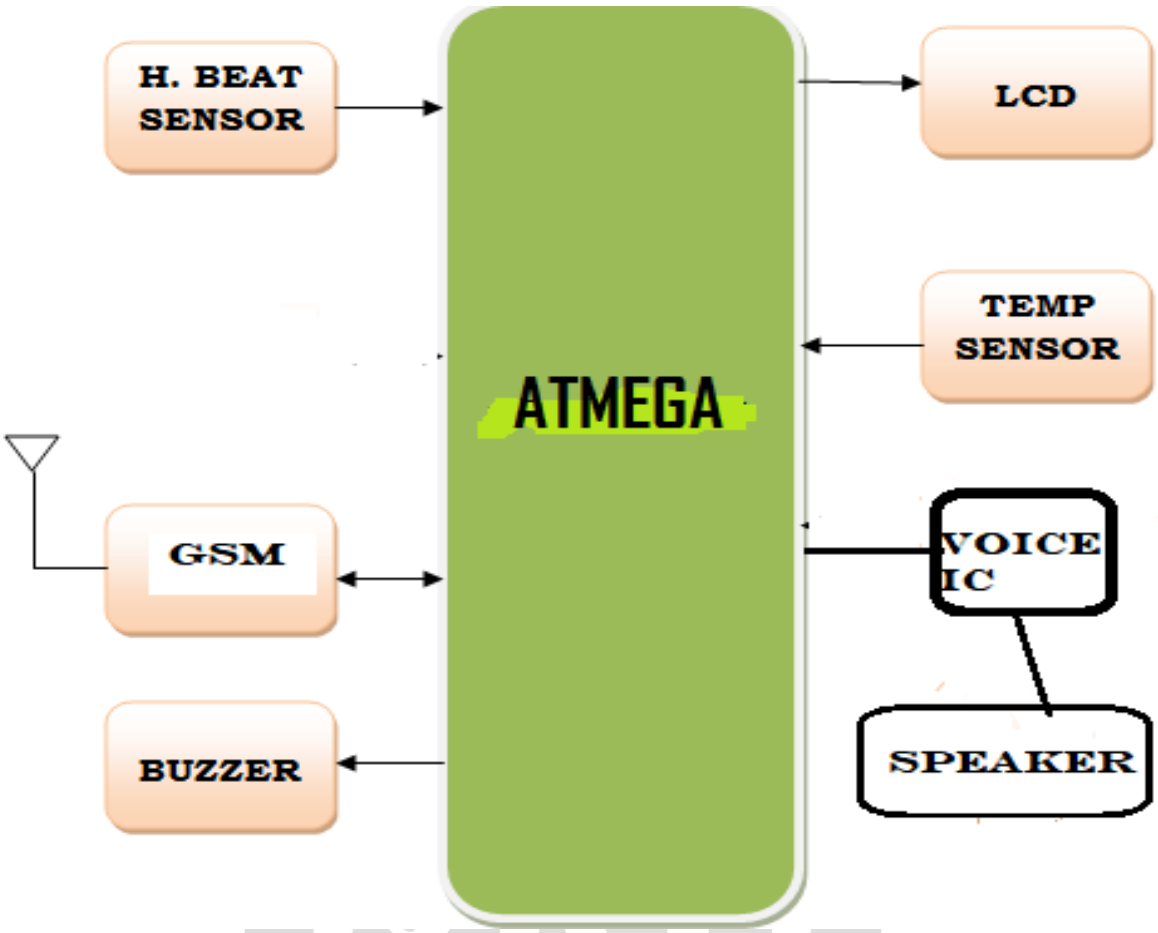


Fig: Block diagram

Working

The diagram represents a system based on an ATMEGA microcontroller, integrating various sensors and modules for monitoring and alert purposes.

Heart Beat Sensor: Measures the heart rate and sends the data to the ATMEGA microcontroller.

Temperature Sensor: Detects body temperature and transmits the information to the ATMEGA.

LCD Display: Shows the measured values (heartbeat, temperature, etc.).

GSM Module: Sends alerts or messages in case of an emergency (e.g., abnormal heart rate or temperature)
Buzzer: Provides an audible alert if an anomaly is detected.

Voice IC and Speaker: Generates voice alerts based on the sensor data.

The ATMEGA microcontroller processes the data from sensors and determines if any abnormal conditions are detected. If so, it triggers appropriate responses like sending alerts via GSM, activating the buzzer, and generating voice messages through the speaker.

HARDWARE COMPONENTS

HARDWARE COMPONENTS

1. Heart Beat Sensor
2. LCD
3. GSM
4. Buzzer
5. Temperature Sensor
6. Jumper wires

Heart Beat Sensor

This heart beat sensor is designed to give digital output of heat beat when a finger is placed inside it .When the heart detector is working, the top-most LED flashes in unison with each heart beat. This digital output can be connected to microcontroller directly to measure the Beats Per Minute (BPM) rate. It works on the principle of light modulation by blood flow through finger at each pulse.

Fig: Heart beat sensor

Features

· Heat beat indication by LED

· Instant output digital signal for directly connecting to microcontroller

· Compact Size

· Working Voltage +5V DC

Applications

· Digital Heart Rate monitor

· Bio-Feedback control of robotics and applications

· Exercise machines

LCD

To display interactive messages we're using LCD Module. We take a look at an smart LCD show of lines, sixteen characters in keeping with line this is interfaced to the controllers. The protocol (handshaking) for the display is as proven. Whereas D0 to D7th bit is the Data strains, RS, RW and EN pins are the control pins and last pins are +5V, -5V and GND to offer supply. Where RS is the Register Select, RW is the Read Write and EN is the Enable pin.The display consists of internal byte-wide registers, one for instructions (RS=zero) and the second for characters to be displayed (RS=1). It additionally includes a user-programmed RAM vicinity (the man or woman RAM) that may be programmed to generate any favored character that can be fashioned the usage of a dot matrix. To distinguish among these two facts regions, the hex command byte 80 could be used to indicate that the display RAM deal with 00h can be selected.Port1 is used to grant the command or statistics type, and ports 3.2 to3.4 provide register pick and study/write ranges.The show takes various quantities of time to perform the features as listed. LCD bit 7 is monitored for common sense high (busy) to ensure the display is overwritten.

Liquid Crystal Display also called as LCD may be very useful in offering user interface in addition to for debugging purpose. The most not unusual sort of LCD controller is HITACHI 44780 which provides a easy interface between the controller & an LCD. These LCD's are quite simple to interface with the controller as well as are price effective.

Fig: LCD

The most commonly used *ALPHANUMERIC* displays are *1x16* (Single Line & 16 characters), *2x16* (Double Line & 16 character per line) & *4x20* (four lines & Twenty characters per line). The LCD requires 3 control lines (RS, R/W & EN) & 8 (or 4) data lines. The number on data lines depends on the mode of operation. If operated in 8-bit mode then 8 data lines + 3 control lines i.e. total 11 lines are required. And if operated in 4-bit mode then 4 data lines + 3 control lines i.e. 7 lines are required. How do we decide which mode to use? It's simple if you have sufficient data lines you can go for 8 bit mode & if there is a time constrain i.e. display should be faster then we have to use 8-bit mode because basically 4-bit mode takes twice as more time as compared to 8-bit mode.

Pin	Symbol	Function
1	Vss	Ground
2	Vdd	Supply Voltage
3	Vo	Contrast Setting
4	RS	Register Select
5	R/W	Read/Write Select
6	En	Chip Enable Signal
7-14	DB0-DB7	Data Lines
15	A/Vee	Gnd for the backlight

Fig: Functions

When RS is low (zero), the statistics is to be handled as a command. When RS is excessive (1), the records being sent is taken into consideration as textual content data which must be displayed on the screen. When R/W is low (0), the facts at the statistics bus is being written to the LCD. When RW is high (1), the program is effectively analyzing from the LCD. Most of the instances there's no need to read from the LCD so this line can immediately be linked to gnd for that reason saving one controller line.

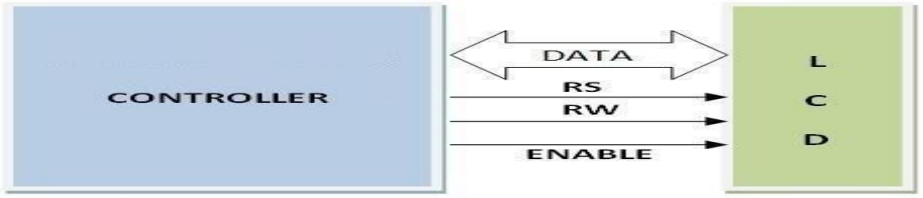


Fig: Controller to LCD

Commands Used In LCD

No.	Instruction	Hex	Decimal
1	Function Set: 8-bit, 1 Line, 5x7 Dots	0x30	48
2	Function Set: 8-bit, 2 Line, 5x7 Dots	0x38	56
3	Function Set: 4-bit, 1 Line, 5x7 Dots	0x20	32
4	Function Set: 4-bit, 2 Line, 5x7 Dots	0x28	40
5	Entry Mode	0x06	6
6	Display off Cursor off (clearing display without clearing DDRAM content)	0x08	8
7	Display on Cursor on	0x0E	14
8	Display on Cursor off	0x0C	12
9	Display on Cursor blinking	0x0F	15
10	Shift entire display left	0x18	24
12	Shift entire display right	0x1C	30
13	Move cursor left by one character	0x10	16
14	Move cursor right by one character	0x14	20
15	Clear Display (also clear DDRAM content)	0x01	1

Fig: Commands in LCD

GSM

The Global System for Mobile Communications (GSM) is a family of standards to describe the protocols for second-generation (2G) digital cellular networks, as used by mobile devices such as mobile phones and mobile broadband modems. GSM is also a trade mark owned by the GSM Association."GSM" may also refer to the voice codec initially used in GSM.

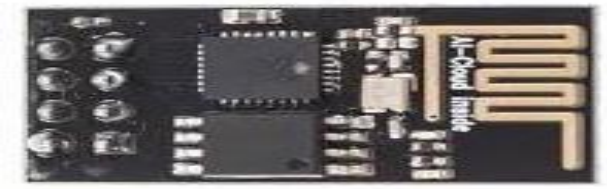


Fig: GSM

2G networks developed as a replacement for first generation (1G) analog cellular networks. The original GSM standard, which was developed by the European Telecommunications Standards Institute (ETSI), originally described a digital, circuit-switched network optimized for full duplex voice telephony, employing time division multiple access (TDMA) between stations. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via its upgraded standards, GPRS and then EDGE. GSM exists in various versions based on the frequency bands used.

GSM was first implemented in Finland in December 1991.[5] It became the global standard for mobile cellular communications, with over 2 billion GSM subscribers globally in 2006, far above its competing standard, CDMA.[6] Its share reached over 90% market share by the mid-2010s, and operating in over 219 countries and territories.[2] The specifications and maintenance of GSM passed over to the 3GPP body in 2000,[7] which at the time developed third-generation (3G) UMTS standards, followed by the fourth-generation (4G) LTE Advanced and the fifth-generation 5G standards, which do not form part of the GSM standard. Beginning in the late 2010s, various carriers worldwide started to shut down their GSM networks; nevertheless, as a result of the network's widespread use, the acronym "GSM" is still used as a generic term for the plethora of G mobile phone technologies evolved from it or mobile phones itself.

BUZZER

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or electronic. Typical uses of buzzers and beepers include alarms, timers and confirmation of user input such as a mouse click or keystroke

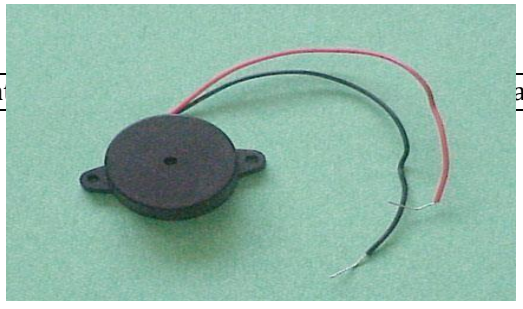


Fig: Buzzer

Features

- The PB series are high-performance buzzers with a unimorph piezoelectric ceramic element and an integral self-excitation oscillator circuit.
- They exhibit extremely low power consumption in comparison to electromagnetic units.
- They are constructed without switching contacts to ensure long life and no electrical noise.
- Compact, yet produces high acoustic output with minimal voltage.

Mechanical

A joy buzzer is an example of a purely mechanical buzzer.

Electromechanical

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

TEMPERATURE SENSOR

The LM35 sensor series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature.

Fig: Temperature sensor

LM35 Sensor Specification

The LM35 series are precision integrated-circuit LM35 temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 sensor thus

has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 sensor does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full -55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to +150°C temperature range, while the LM35C sensor is rated for a -40° to +110°C range (-10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D sensor is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

LM35 Sensor Circuit Schematic

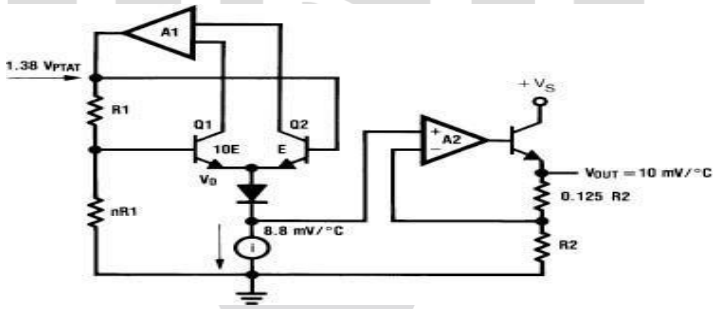


Fig: LM35 Circuit diagram

LM35 Sensor Pin outs and Packaging

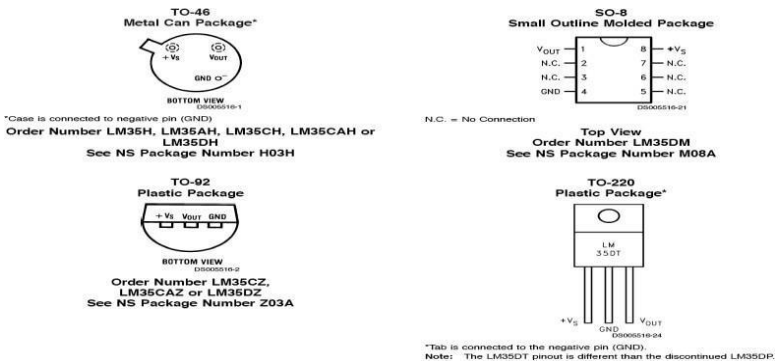


Fig: LM35 packaging

LM35 Sensor Sources

There are several manufacturers of this popular part and each has LM35 sensor specs, datasheets and other free LM35 downloads. This amplifier is available from the following manufacturers.

- ✓ National Semiconductor
- ✓ On Semiconductor
- ✓ Texas Instruments
- ✓ Fairchild Semiconductor
- ✓ STMicroelectronics
- ✓ Jameco Electronics
- ✓ Analog Devices

JUMPER WIRES

Jumper wires are simple electrical wires used to make connections between different components in a circuit. They are essential for building and prototyping electronic projects. Here's a breakdown of their features and uses:

Jumper wires typically have a connector on each end, allowing them to easily plug into breadboards, microcontrollers, or other electronic devices. They come in various lengths, colors, and types.

The most common types are:

Male-to-Male: Both ends have male connectors (pins) that fit into female headers on components or breadboards.

Female-to-Female: Both ends have female connectors that can accept male pins, useful for connecting to male headers.

Fig: Jumper wires

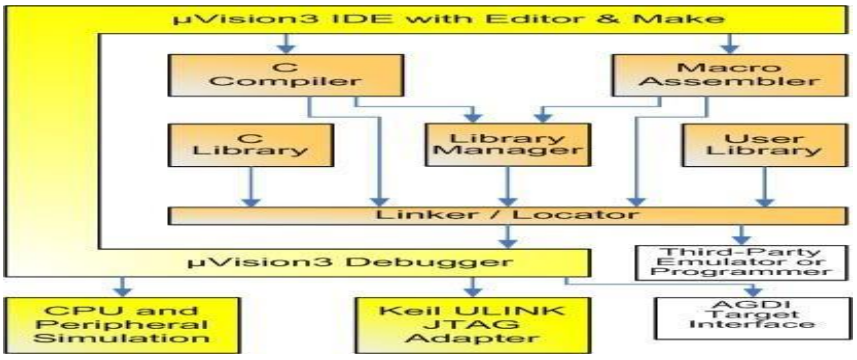
Male-to-Female: One end has a male connector and the other end has a female connector, allowing you to connect a male pin to a female header.

Using jumper wires is straightforward: you simply plug one end into a pin on a microcontroller, sensor, or breadboard, and the other end into the corresponding pin of another component. This allows you to create connections for power, ground, and data signals in your circuit. They are especially useful in prototyping because they enable quick and easy modifications to your circuit without soldering, making it simple to test different configurations and components

DESIGN SOFTWARE

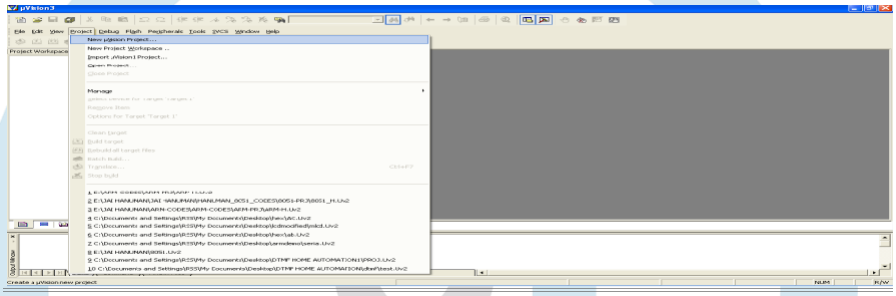
Introduction:

In this chapter the software used and the language in which the program code is defined is mentioned and the program code dumping tools are explained. The chapter also documents the development of the program for the application. This program has been termed as “Source code”. Before we look at the source code we define the two header files that we have used in the code.

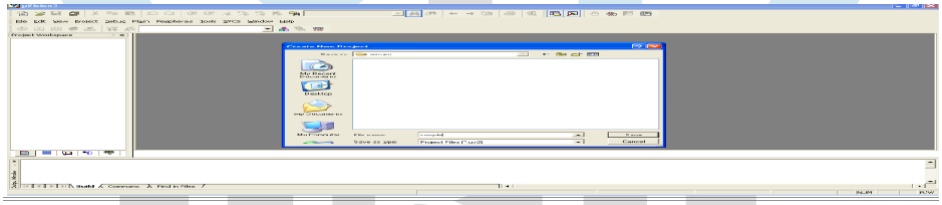


Keil development tools for the LPC2148 Microcontroller Architecture support every level of software developer from the professional applications

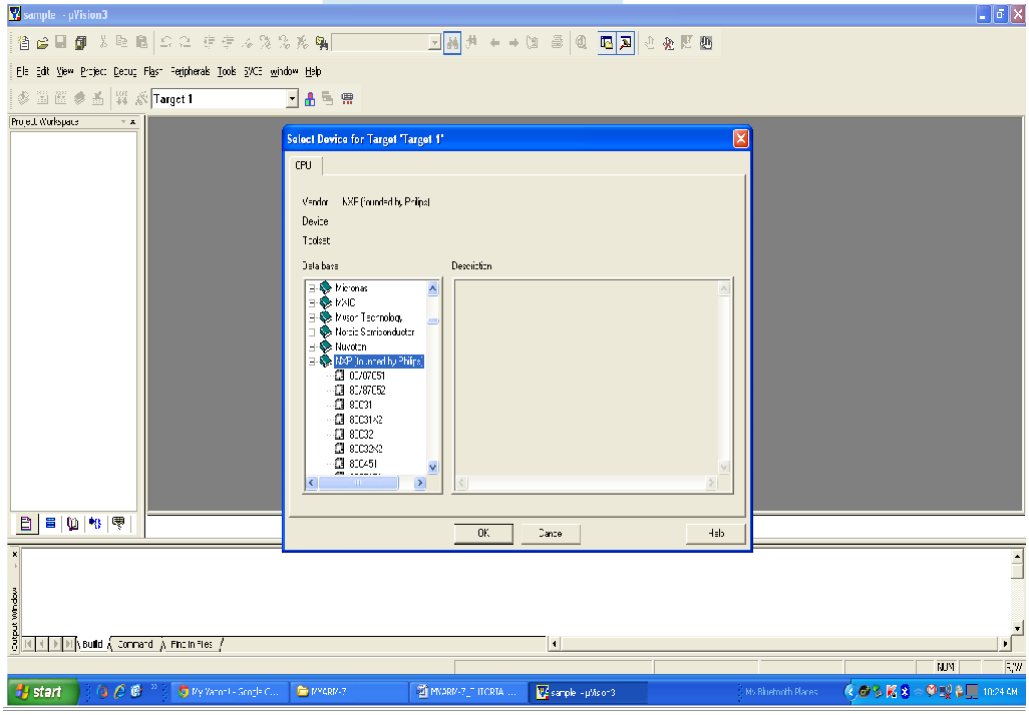
Step1: open the NEW U version project which is in the project option in the tool bar.



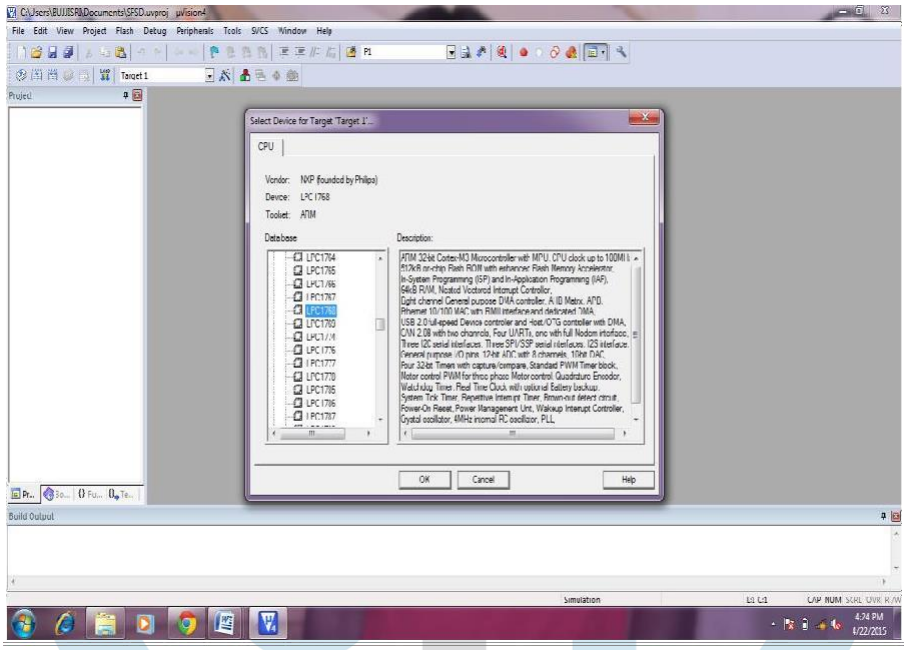
Step2: save the project with the required name and click save button.



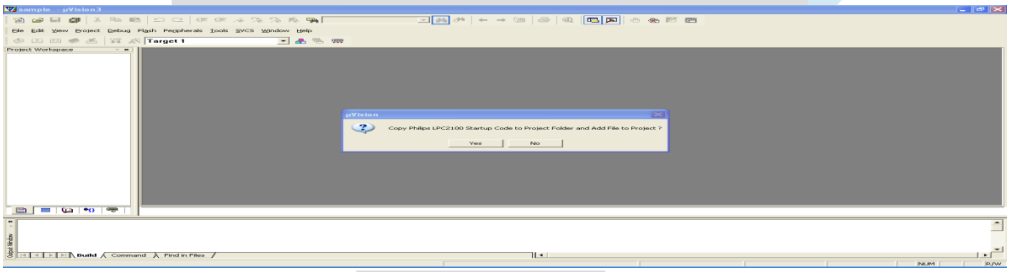
Step:3 select the device from NXP options the window which comes after saving the project.



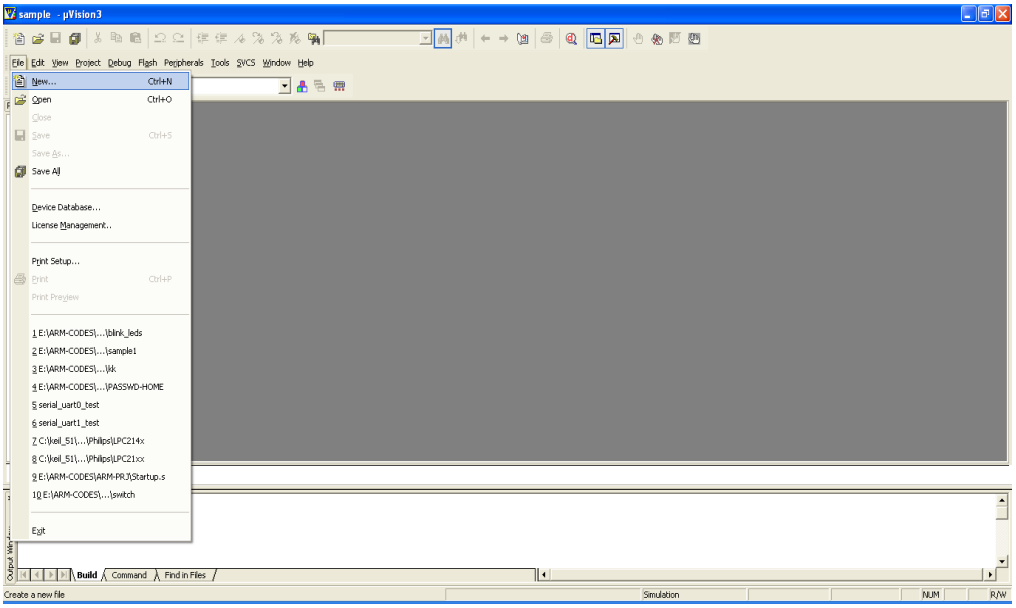
Step 4: select the LPC 1768 from the NXP options



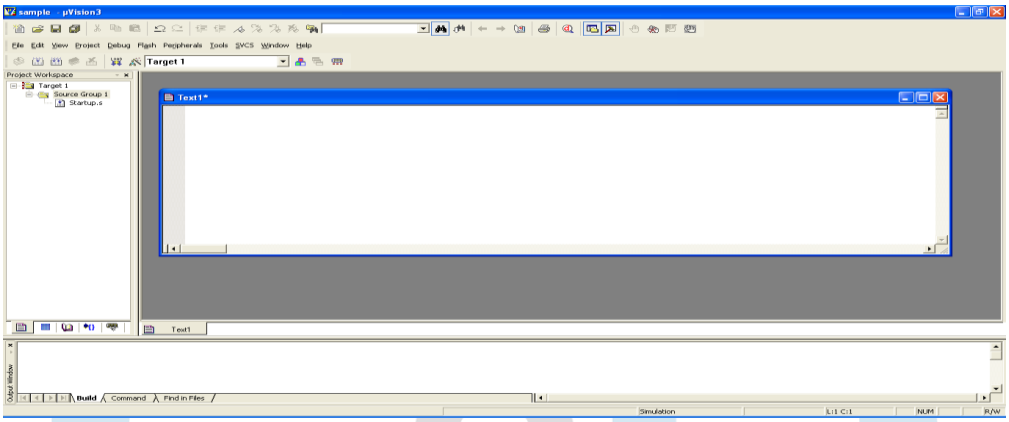
Step 5: click yes in the startup code of LPC 1768 window



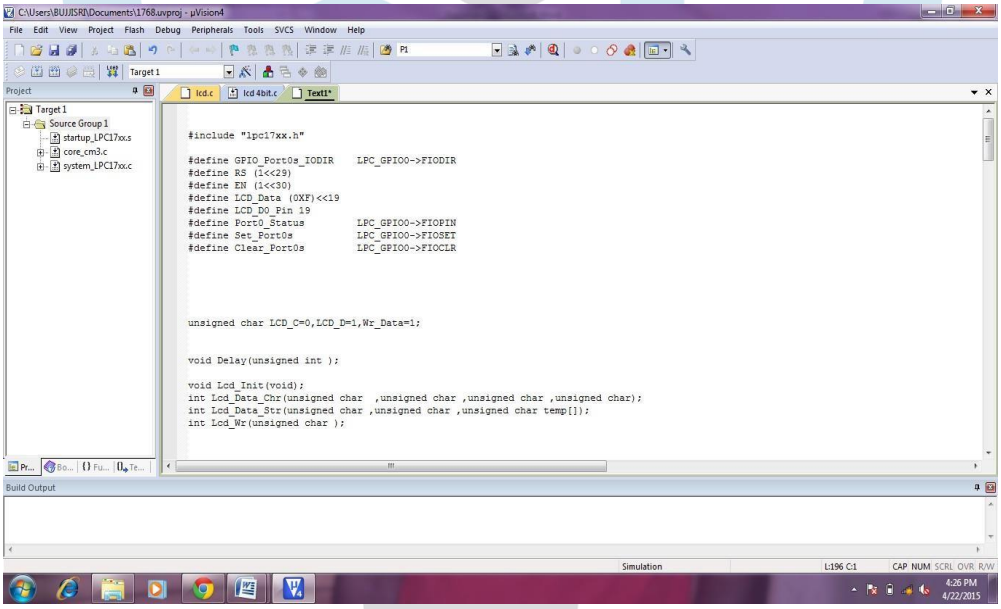
Step 6: select the new file from the file options to write the C code



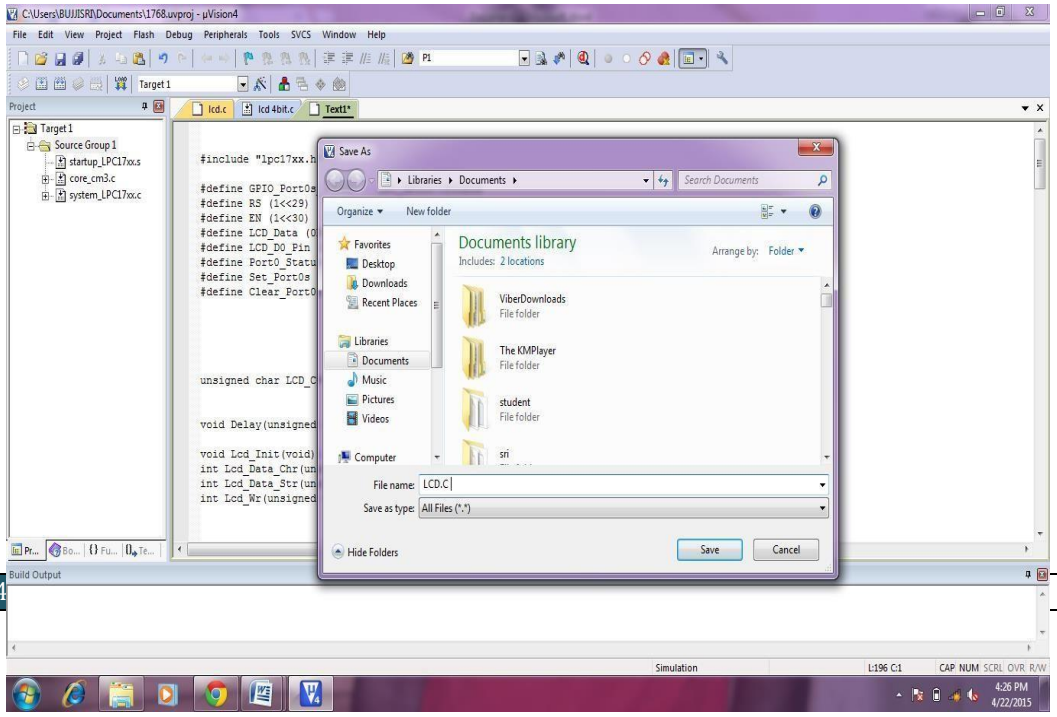
Step 7: a new text file will open which we have to write the following code



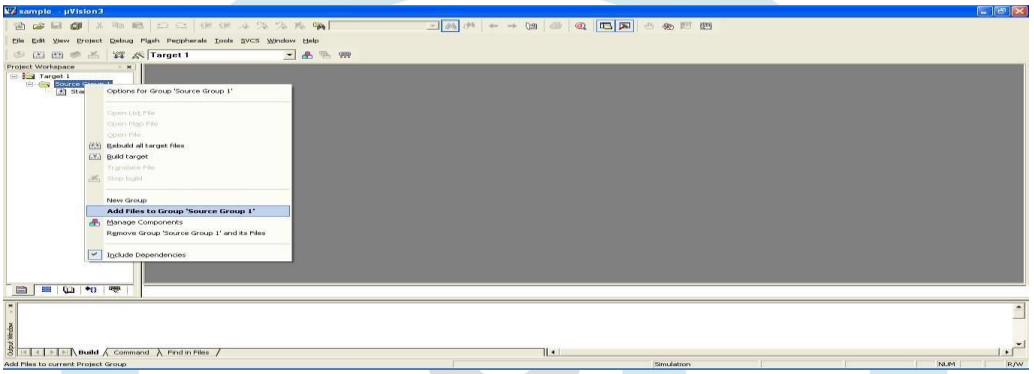
Step 8 : write the code require for ur applications



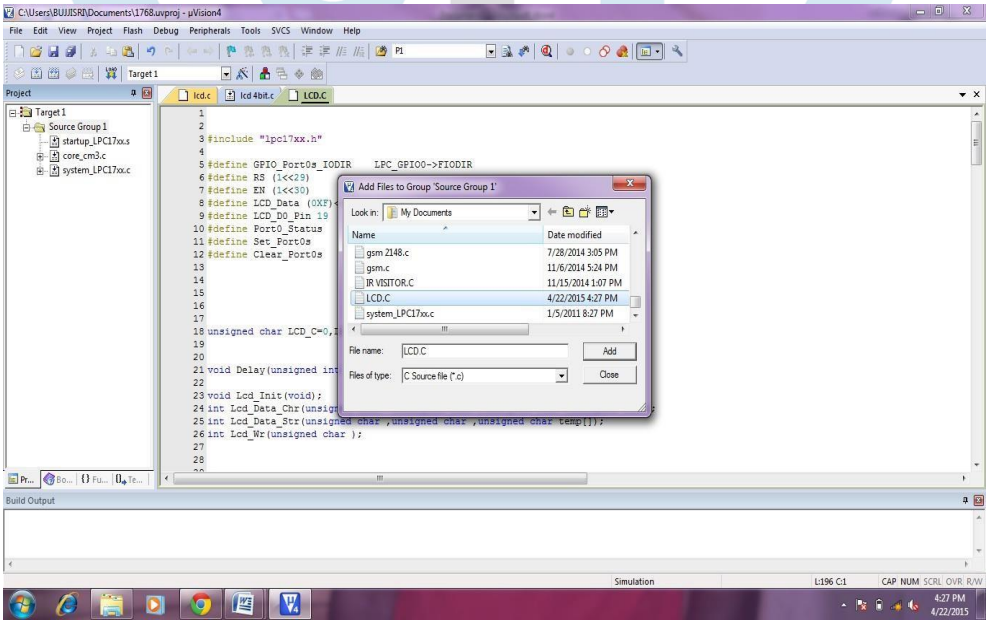
Step 9: Save the text file as .C file



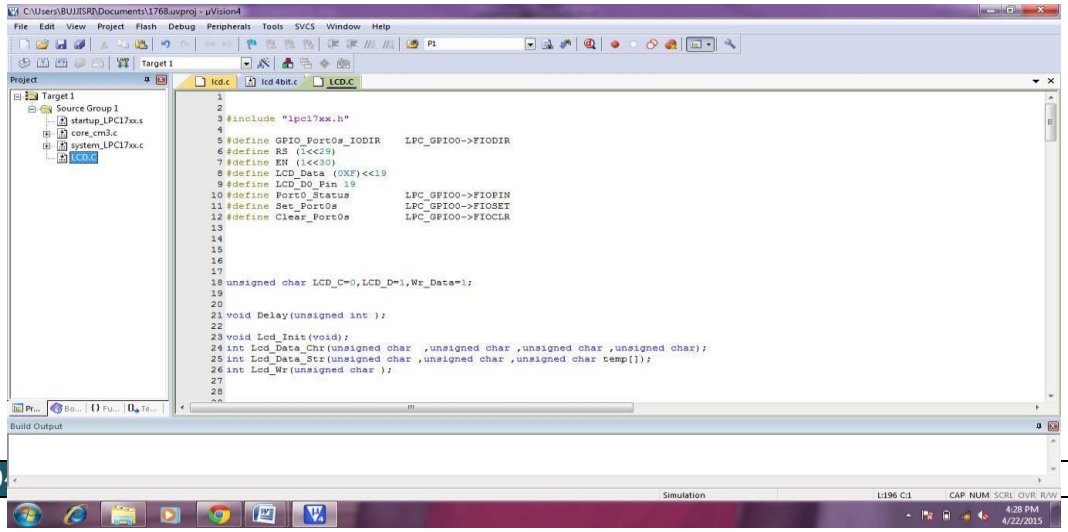
Step 10: Add the saved .C file p by right clicking the Source Group Option1 and u will get drop down window in that select Add Files to Group



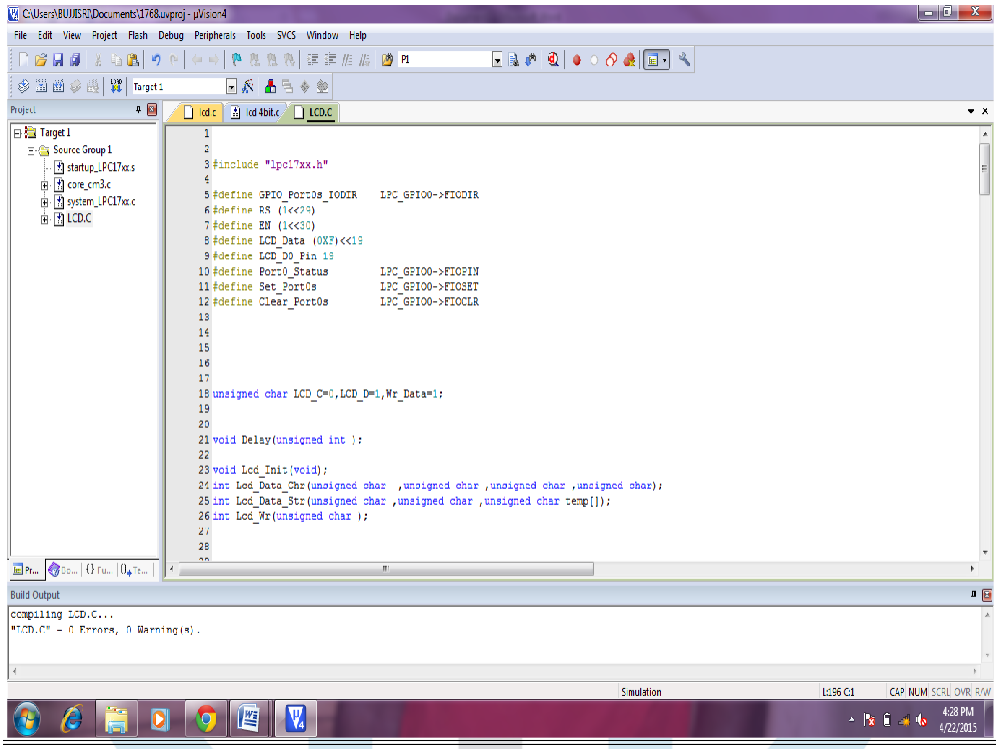
Step 11: select the saved .c file from the window which is showing to add the file by clicking the add tab.



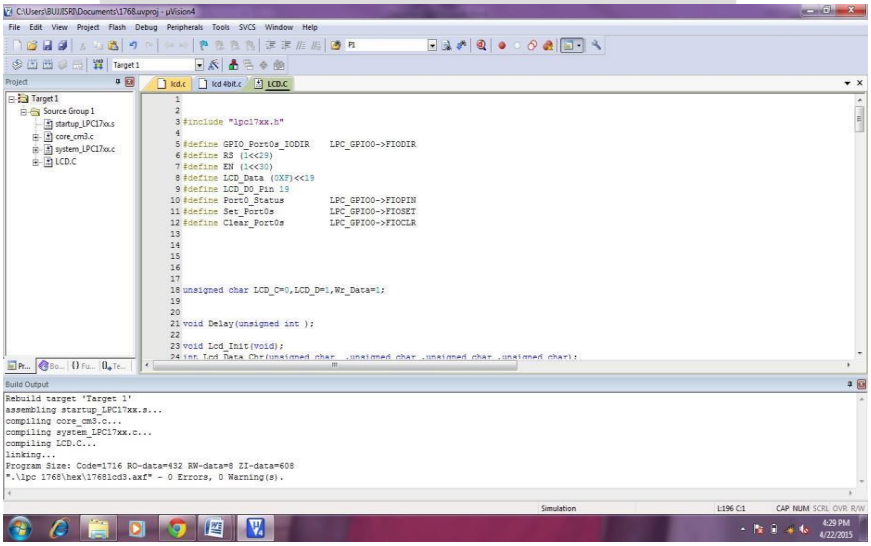
Step 12: Thus .C file is added to the Source Group is clearly seen

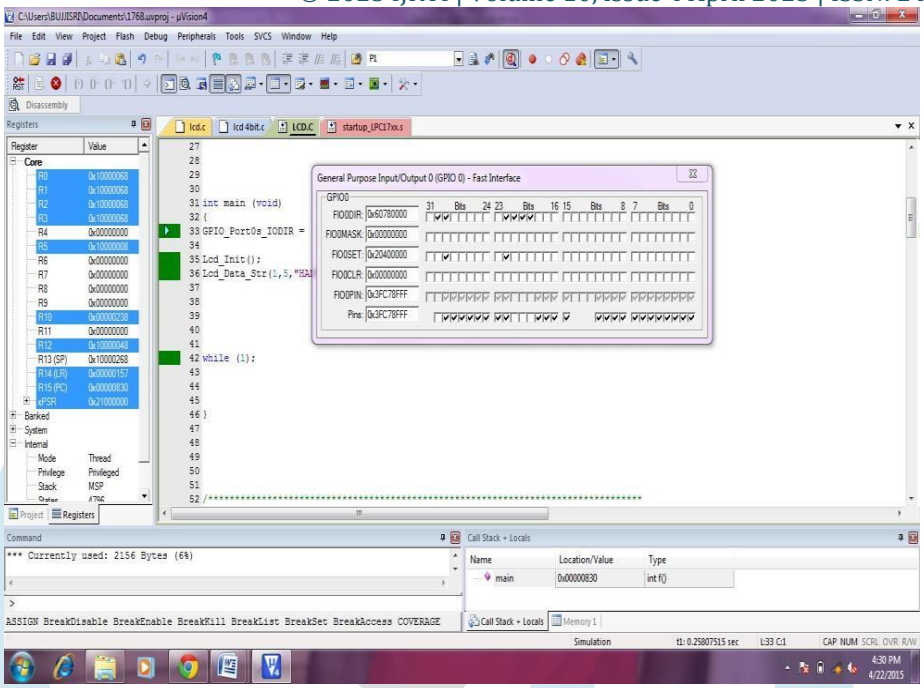


Step 13: Press the icon that shows Rebuilt Target which proceeds for the Linking if the .C file.



Step 14: Re-built All Target files for complete process of assembling->Compiling ->linking-> generating .hex file, press the start/stop debug icon for debugging of the code which written. Following are debugging windows press F11 for step by step debug Window:1





Advantages

1. **Enhanced Situational Awareness:** Soldiers receive real-time updates on enemy movements, terrain conditions, and mission objectives. Embedded GPS tracking allows commanders to monitor soldier locations and plan strategies accordingly.

2. **Health Monitoring & Injury Detection:** Wearable biosensors can track vital signs (heart rate, temperature, hydration, etc.), detecting fatigue or injuries. Automated alerts notify medics when a soldier is in distress or unconscious.

3. **Improved Communication & Coordination:** Wireless networks ensure seamless communication between soldiers, drones, and command centers. Enables secure, encrypted data transmission to prevent enemy interception.

4. **Threat Detection & Early Warning Systems:** Smart helmets and body armor with embedded sensors can detect gunfire, explosions, or toxic gases. AI-powered systems can analyze threats and provide real-time alerts for evasive action.

5. **Lightweight & Power-Efficient Equipment:** Embedded electronics are designed to be compact, energy-efficient, and rugged for military environments. Reduces the need for bulky, wired systems, increasing soldier mobility.

6. **Autonomous Support & Decision-Making:** AI-driven systems can assist soldiers by analyzing battlefield data and providing tactical recommendations .Helps in identifying safe routes, potential ambush points, and strategic cover positions.

7. **Remote Weapon & Drone Integration:** Soldiers can control autonomous drones and remotely operated weapons using embedded wireless systems. Enhances surveillance, reconnaissance, and attack capabilities without direct exposure to

danger.

8. **Smart Ammunition & Inventory Management:** Embedded RFID and IoT sensors track ammunition, medical supplies, and rations in real time. Ensures soldiers never run out of critical resources during missions.

9. **Cyber security & Jamming Resistance:** Advanced encryption and anti-jamming techniques ensure secure communication in electronic warfare environments. Real-time threat detection prevents hacking or signal interference by adversaries.

Disadvantages

1. **Cyber security Vulnerabilities:** Wireless systems are susceptible to hacking, jamming, and signal interception by enemy forces.If compromised, real-time data leaks could expose troop positions and mission strategies.

2. **Power & Battery Limitations:** Embedded devices require continuous power, and battery life is often limited. Frequent charging or battery replacements can be impractical in combat situations.

3. **Signal Interference & Connectivity Issues:** Harsh battlefield environments may cause signal loss or disruptions (e.g., mountainous regions, urban warfare). Enemies could deploy electronic warfare (EW) tactics to jam or block communications.

4. **Increased Complexity & Training Requirements:** Soldiers need specialized training to operate and maintain these advanced systems. The learning curve can be steep, especially for non-tech-savvy personnel.

5. **Risk of Over-Reliance on Technology:** Dependence on real-time data may reduce a soldier's decision-making abilities in case of system failure. If AI-driven recommendations are incorrect, soldiers may follow flawed tactics without questioning them.

6. **High Cost & Maintenance Challenges:** Advanced wireless embedded systems are expensive to develop, deploy, and maintain. Repairs or replacements in remote or battlefield conditions can be logistically difficult.

7. **Increased Weight & Equipment Burden:** Despite efforts to make systems lightweight, wearable devices, batteries, and communication modules add extra

weight. Heavy gear can lead to reduced mobility, fatigue, and slower response times.

Applications

1. **Real-Time Soldier Health Monitoring:** Wearable biosensors track vital signs (heart rate, body temperature, hydration, stress levels). Detects injuries, fatigue, or unconsciousness, alerting medics in real time. Helps prevent heatstroke, dehydration, or cardiac arrest during combat.

2. **GPS & Location Tracking:** Real-time GPS tracking allows commanders to monitor soldier movements. Helps in coordinating tactical strategies, rescue missions, and avoiding friendly fire Essential for search-and-rescue operations in hostile environments.

3. **Wireless Communication Systems:** Secure radio and satellite communication networks keep soldiers connected.AI-assisted battlefield chatbots provide automatic translations for multilingual operations. Enables real-time mission updates and emergency alerts without delays.

4. **Smart Helmets & Augmented Reality (AR) Systems:** AR-based smart helmets display real-time battlefield data (maps, enemy locations, health stats). Integrated thermal vision and night vision enhance visibility in low-light conditions. Helps soldiers navigate complex terrains without distractions.

5. **Embedded Threat Detection Sensors:** Smart suits with ballistic impact sensors detect enemy gunfire or explosions. Sensors can identify chemical, biological, radiological, or nuclear (CBRN) threats. AI algorithms analyze gunshot sounds, drone activity, and movement patterns for early warnings.

6. **Drone & Robotics Integration:** Soldiers can deploy wireless-controlled drones for reconnaissance and surveillance. Unmanned ground vehicles (UGVs) provide real-time battlefield assessments. Reduces risk by allowing robotic systems to engage threats remotely.
7. **AI-Based Decision Support Systems:** AI-powered analytics suggest optimal movement paths, cover points, and escape routes. Helps soldiers make quick, data- driven decisions under pressure. Machine learning improves threat prediction based on historical combat data.

8. **Smart Ammunition & Inventory Tracking:** RFID and IOT sensors monitor ammo levels, medical supplies, and food rations. Ensures soldiers never run out of critical resources during missions. Automates logistics and reduces human error in inventory management.

9. **Autonomous Medical Evacuation & Combat Assistance:** Injured soldiers can trigger automated distress signals with real-time location data. Autonomous medevac drones or robotic stretchers can retrieve and transport wounded personnel. Reduces casualties by speeding up evacuation and first aid response times.

10. **Cyber security & Encrypted Communication:** End-to-end encryption protects soldier communication from hacking or interception. AI-driven cyber threat detection identifies potential electronic warfare attacks. Advanced jamming-resistant networks prevent signal disruptions from enemies.

Algorithm

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4,
3, 2); #include
<SoftwareSerial.h>
float
pulse =
0; float
temp =
0;
SoftwareSerial ser(9,10);
String apiKey = "OO707TGA1BLUNN12";


// Variables
int pulsePin = A0; // Pulse Sensor purple wire connected to analog
pin 0 int blinkPin = 7 ; // pin to blink led at each beat
```

```
int fadePin = 13; // pin to do fancy classy fading blink at each

beat int fadeRate = 0; // used to fade LED on with PWM on

fadePin



lcd.clear();

lcd.setCursor(0,0);

lcd.print(" Patient

Health");

lcd.setCursor(0,1);

lcd.print(" Monitoring

"); delay(4000);

lcd.clear();


lcd.setCursor(0,0);

lcd.print("Initializin

g..... ");

delay(5000);

lcd.clear();

lcd.setCursor(

0,0);

lcd.print("Getting Data ... ");

ser.begin(9600);

ser.println("AT");

delay(1000);

ser.println("AT+GMR

"); delay(1000);

ser.println("AT+CWMODE
```

```
=3"); delay(1000);
ser.println("AT+RST");
delay(5000);
ser.println("AT+CIPMUX=
1"); delay(1000);

String cmd="AT+CWJAP=\"Alexahome\",\"98765432\"";
ser.println(cmd);
delay(1000);
ser.println("AT+CIFSR");
delay(1000);
}
// Where the Magic
Happens void loop()
{
serialOutput();
if (QS == true) // A Heartbeat Was Found
{

// BPM and IBI have been Determined
}
ledFadeToBeat(); // Makes the LED Fade Effect Happen
delay(20); // take a break
read_t
emp();
esp_82
66();
}
```

```
void ledFadeToBeat()

{

fadeRate -= 15; // set LED fade value

fadeRate = constrain(fadeRate,0,255); // keep LED fade value from going into negative numbers!

analogWrite(fadePin,fadeRate); // fade LED

}

void interruptSetup()

{
// Initializes Timer2 to throw an interrupt every 2mS.

TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE

TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER

OCR2A = 0X7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE

RATE TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2

AND OCR2A sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED

}

void serialOutput()

{ // Decide How To Output

Serial. if (serialVisual ==

true)

{

arduinoSerialMonitorVisual('-', Signal); // goes to function that makes Serial Monitor Visualizer

}

else

{

sendDataToSerial('S', Signal); // goes to sendDataToSerial function

}

}
```

```
void serialOutputWhenBeatHappens()
{
if (serialVisual == true) // Code to Make the Serial Monitor Visualizer Work
{
Serial.print("*** Heart-Beat Happened *** "); //ASCII Art Madness
Serial.print("BPM: ");
Serial.println(BPM);
}
else
{
sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix
sendDataToSerial('Q',IBI); // send time between beats with a 'Q'
prefix
}
}
void arduinoSerialMonitorVisual(char symbol, int data )
{
const int sensorMin = 0; // sensor minimum, discovered through
experiment const int sensorMax = 1024; // sensor maximum, discovered
through experiment int sensorReading = data; // map the sensor range to a
range of 12 options:
int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
// do something different depending on the
// range value:
switch (range)
{
case 0:
Serial.println(""); /////ASCII Art
```

```
Madness break;

case 1: Serial.println("---");

break; case 2:

Serial.println("------ ");

break; case 3:

Serial.println("--------- ");

break; case 4:

Serial.println("---------- ");

break; case 5:

Serial.println("---------- |-");

break; case 6:

Serial.println("----------|----");

break; case 7:

Serial.println("----------|--------");

break;

case 8:

Serial.println("----------|----------");

break;

case 9:

Serial.println("----------|---------- ");
break;

case 10:

Serial.println("----------|----------");

break;

case 11:

Serial.println("----------|---------- ");

break;
```

```
}
}

void sendDataToSerial(char symbol, int data )
{
Serial.print(symbol);
Serial.println(data);
}
ISR(TIMER2_COMPA_vect) //triggered when Timer2 counts to 124
{
cli(); // disable interrupts while we do this
Signal = analogRead(pulsePin); // read the Pulse Sensor
sampleCounter += 2; // keep track of the time in mS with this
variable
int N = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise
// find the peak and trough of the pulse wave

if(Signal < thresh && N > (IBI/5)*3) // avoid dichrotic noise by waiting 3/5 of last IBI
{
if (Signal < T) // T is the trough
{
T = Signal; // keep track of lowest point in pulse wave
}
}
if(Signal > thresh && Signal > P)
{ // thresh condition helps avoid
noise P = Signal; // P is the
peak
```

```
} // keep track of highest point in pulse wave

// NOW IT'S TIME TO LOOK FOR THE HEART BEAT

// signal surges up in value every time there is a

pulse if (N > 250)

{ // avoid high frequency noise

if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )

{

Pulse = true; // set the Pulse flag when we think there is a pulse

digitalWrite(blinkPin,HIGH); // turn on pin 13 LED

IBI = sampleCounter - lastBeatTime; // measure time between beats in mS

lastBeatTime = sampleCounter; // keep track of time for next pulse


if(secondBeat)

{ // if this is the second beat, if secondBeat == TRUE

secondBeat = false; // clear secondBeat flag

for(int i=0; i<=9; i++) // seed the running total to get a realisitic BPM at startup

{

rate[i] = IBI;

}

}

if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE

{

firstBeat = false; // clear firstBeat flag

secondBeat = true; // set the second beat

flag sei(); // enable interrupts again

return; // IBI value is unreliable so discard it

}

// keep a running total of the last 10 IBI values
```

```
word runningTotal = 0; // clear the runningTotal variable
for(int i=0; i<=8; i++)
{ // shift data in the rate array
rate[i] = rate[i+1]; // and drop the oldest IBI value
runningTotal += rate[i]; // add up the 9 oldest IBI values
}
rate[9] = IBI; // add the latest IBI to the rate array
runningTotal += rate[9]; // add the latest IBI to
runningTotal runningTotal /= 10; // average the last 10
IBI values
BPM = 60000/runningTotal; // how many beats can fit into a minute? that's
BPM! QS = true; // set Quantified Self flag
// QS FLAG IS NOT CLEARED INSIDE THIS ISR
pulse = BPM;
}
}
if (Signal < thresh && Pulse == true)
{ // when the values are going down, the beat is
over digitalWrite(blinkPin,LOW); // turn off pin
13 LED Pulse = false; // reset the Pulse flag so we
can do it again amp = P - T; // get amplitude of the
pulse wave
thresh = amp/2 + T; // set thresh at 50% of the
amplitude P = thresh; // reset these for next time
T = thresh;
}
if (N > 2500)
```

```
{ // if 2.5 seconds go by without
a beat thresh = 512; // set thresh
default
P = 512; // set P
default T = 512;
// set T default
lastBeatTime = sampleCounter; // bring the lastBeatTime up to date
firstBeat = true; // set these to avoid noise
secondBeat = false; // when we get the heartbeat back
}
sei(); // enable interrupts when youre done!
}// end isr
void esp_8266()
{
String cmd = "AT+CIPSTART=4,\"TCP\",\"";
cmd += "184.106.153.149"; //
api.thingspeak.com cmd += "\",80";
ser.println(c
md);
Serial.println(
cmd);
if(ser.find("E
rror"))
{
Serial.println("AT+CIPSTART
error"); return;
}
String getStr = "GET
```

```
/update?api_key="; getStr += apiKey;

getStr

+="&field1=";

getStr

+=String(temp);

getStr

+="&field2=";

getStr

+=String(pulse);

getStr += "\r\n\r\n";

// send data length

cmd = "AT+CIPSEND=4,";

cmd +=

String(getStr.length());

ser.println(cmd);

Serial.println(cmd);

delay(100

0);

ser.print(g

etStr);

Serial.println(getStr); //thingspeak needs 15 sec delay between

updates delay(3000);

}

void read_temp()

{

int temp_val =

analogRead(A1); float mv =

(temp_val/1024.0)*5000;
```

```
float cel = mv/10;

temp = (cel*9)/5 + 32;

Serial.print("Temperature:"

); cd.setCursor(0,0);

lcd.print("BPM :");

lcd.setCursor(7,0);

lcd.print(BPM);

lcd.setCursor(0,1);

lcd.print("Temp.:");

lcd.setCursor(7,1);

lcd.print(temp);

lcd.setCursor(13,1);

lcd.print("F")
```

RESULT

Recent advancements in wireless embedded electronics have significantly enhanced soldier security by integrating real-time health monitoring, precise location tracking, and efficient communication systems. These technologies collectively improve situational awareness and expedite emergency responses.

CONCLUSION

The project has been successfully designed and tested. Integrating features of all the hardware components used have developed it. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. Secondly, using highly advanced IC's and with the help of growing technology the project has been successfully implemented.

FUTURE SCOPE

The future of real-time wireless embedded electronics for soldier security will likely witness a convergence of various cutting-edge technologies like AI, IoT, quantum communications, and wearable health tech. These advancements will not only enhance

the safety, health, and efficiency of soldiers but also redefine how military operations are conducted, making them more agile, responsive, and technologically integrated.

REFERENCES

1. Scarponi G E, Landucci G, Heymes F and Cozzani V 2018 Experimental and numerical study of the behavior of LPG tanks exposed to wildland fires Process Safety and Environmental Protection

2. Sabiq A and Alfarisi T 2018 Sistem Wireless Sensor Network Berbasis Arduino Uno dan Raspberry Pi untuk Pemantauan Kualitas Udara di Cempaka Putih Timur, Jakarta Pusat 301– 306.

3 Dian S K 2012 Analisis Konsekuensi Dispersi Gas, Kebakaran, dan Ledakan Akibat Kebocoran Tabung Lpg 12 Kg Di Kelurahan Manggarai Selatan Tahun 2012 Dengan Menggunakan Breeze Incident Analyst Software Selama (Depok: Universitas Indonesia)

4. Tian Y, Lv Y and Tong L 2013 Design and application of sink node for wireless sensor network COMPEL-The international journal for computation and mathematics in electrical and electronic engineering 32(2) 531-544

5. Kim 2014 MQ-6 LPG LNG Gas Sensor Module _ Sandbox Electronics Sandbox Electronics [Online] Available: http://sandboxelectronics.com/?p=191 [Accessed: 01-Jan-2017]

6. Bedell F and Ryan H J 1895 Action or a single-phase synchronous motor Journal of the Franklin Institute 139(3) 197-214

7. Arduino LLC 2017 Arduino Nano 9210662 1

8. Data T 2014 MQ-6 Semiconductor Sensor for LPG 2–4

9. Dhawale D S, Salunkhe R R, Patil U M, Gurav K V, More A M and Lokhande C D 2008 Room temperature liquefied petroleum gas (LPG) sensor based on p-polyaniline/n-TiO2 heterojunction Sensors and Actuators B: Chemical 134(2) 988-992