



*Ministry of higher Educations And Scientific
research
Middle Technical University
Kut Technical institute
Electrical department*



DESIGN AND IMPLEMENTATION OF Smart Hart Rate Measurement

A project submitted in Partial
Fulfillment of the Requirement for
Degree of Diploma In Electrical

By

Sajad Fadel

Hussain Taghi

Supervised by

Bahaa Kareem

2019 – 2020

DEPARTURMENT OF ELECTRICAL TECHNIQUES
KUT TECHNICAL INSTITUTE

Certificate

This is certify that the project worked entitled “ DESIGN AND IMPLEMENTATION OF Smart Hart Rate Measurement " by Sajad Fadel Hussain Taghi are student of department of electrical techniques , kut technical intitute in partial fulfillment of the requirements for the award of the degree of Bachelor of Techniques In Electrical

The Guide

Head of the Department

DECLARATION

I declare that the work reported in the project entitled " DESIGN AND IMPLEMENTATION OF Smart Hart Rate Measurement " is record of the work done by our in the department of electrical techniques , kut technical intitute .

ACKNOWLEDGMENT

First and foremost, we would like to extend my sincere thanks to our guide Department of Electrical Techniques , for his guidance , encouragement , motivation and continued support throughout our project work. He has allowed to pursue our project interests with sufficient freedom, while always being there to guide ours. Working with him has been one of the most rewarding experiences of our professional life.

Also, we would like to extend my thanks to Ass. Professor and Head, department of electrical techniques , kut technical intitute for their supporting and continuing help. We would like to thank all our teachers and all the staff of department of electrical techniques , kut technical intitute for their supporting and continuing help .

We would like to thank our fellow project mates for helping me and sharing their knowledge throughout the project duration , and

Finally , we wish to thank all those people who help our and all our families members for their supported in successful completion of studying in Department of Electrical Techniques .

DEDICATION

To the lighthouse science, Great Prophet and Human Teacher

Muhammad peace be upon him and his family .

To our homeland (IRAQ)..

To our parents...

To our brothers, to our families, and

To all who help our

1.1 : Introduction

The heart is one of the most vital organs within the human body. It acts as a pump that circulates oxygen and nutrient carrying blood around the body in order to keep it functioning. The circulated blood also removes waste products generated from the body to the kidneys. When the body is exerted the rate at which the heart beats will vary proportional to the amount of effort being exerted. By detecting the voltage created by the beating of the heart, its rate can be easily observed and used for a number of health purposes

Heartbeat and body temperature are the major signs that are routinely measured by physicians after the arrival of a patient. Heart rate refers to how many times a heart contracts and relaxes in a unit of time (usually per minute). Heart rate varies for different age groups. For a human adult of age 18 or more years, a normal resting heart rate is around 72 beats per minute (bpm). The functioning of heart can be called as efficient if it is having lower heart rate when the patient is at rest. Babies have a much higher rate than adults around 120 bpm and older children have heart rate around 90 bpm.

So this research describes the design of a very low-cost patient monitoring system which measures heart rate of a patient by using arduino and pulse sensor , The arduino uno processes this data and displays it in LCD, and physician or doctor will be able to examine him/her. This device will be much needed during emergency period or for saving time of both patient and doctor.

1.2 : Tools used in the project

- 1- Arduino Board .
- 2- Pulse Sensor .
- 3- LCD I2C.
- 4- Wire .
- 5- Breadboard

1.2.1 : Arduino Board

Is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board often referred to as a microcontroller and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

The Arduino hardware and software was designed for artists, designers, hobbyists, hackers, student, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smart-phone or your TV! This flexibility combined with the fact that the Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a huge variety of Arduino-based projects

1.2.1.1 : Type of Arduino

There are a several type of Arduino board Fig. 1.1 , the mean different between it are the speed of processor and the number of pins , The most common typical are :

- Arduino Nano

- Arduino Mini
- Arduino micro
- Arduino Leonardo
- Arduino Uno
- Arduino Mega



Figure 1.1 : Arduino Boards

We will talk about Arduino UNO only . because we used it in this Project

1.2.1.2 : Arduino Uno

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worrying too

much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. Figure 1.2 & 1.3 .

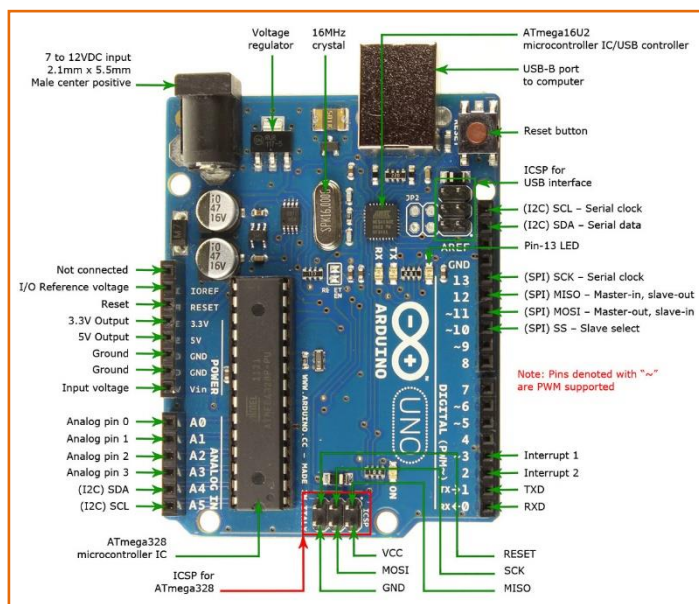


Fig. 1.2 : Arduino UNO Pins

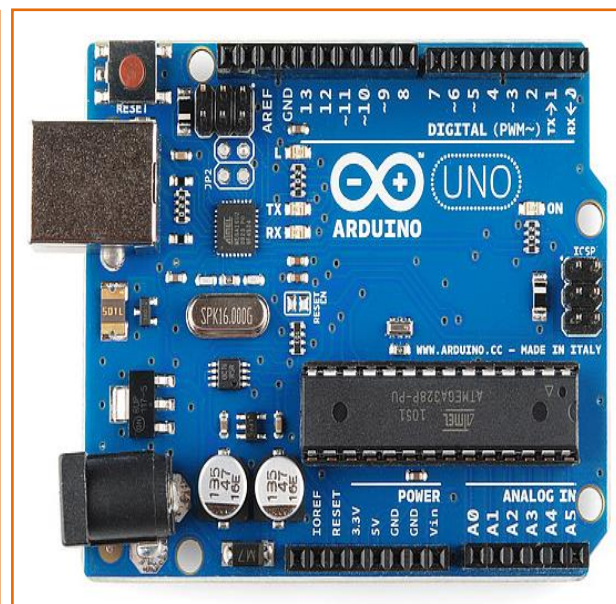


Fig. 1.3 : Arduino UNO Board

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards and datasheet see the last chapter .

We can transfer the code from the computer to the Arduino through a USB cable as shown in the figure 1.4



Fig.1.4 : USB cable of Arduino UNO

1.2.2 : Pulse Sensor

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heartrate data into their projects. The sensor clips onto a fingertip or earlobe and plugs right into Arduino with some jumper cables. It also includes an open-source monitoring app that graphs your pulse in real time.

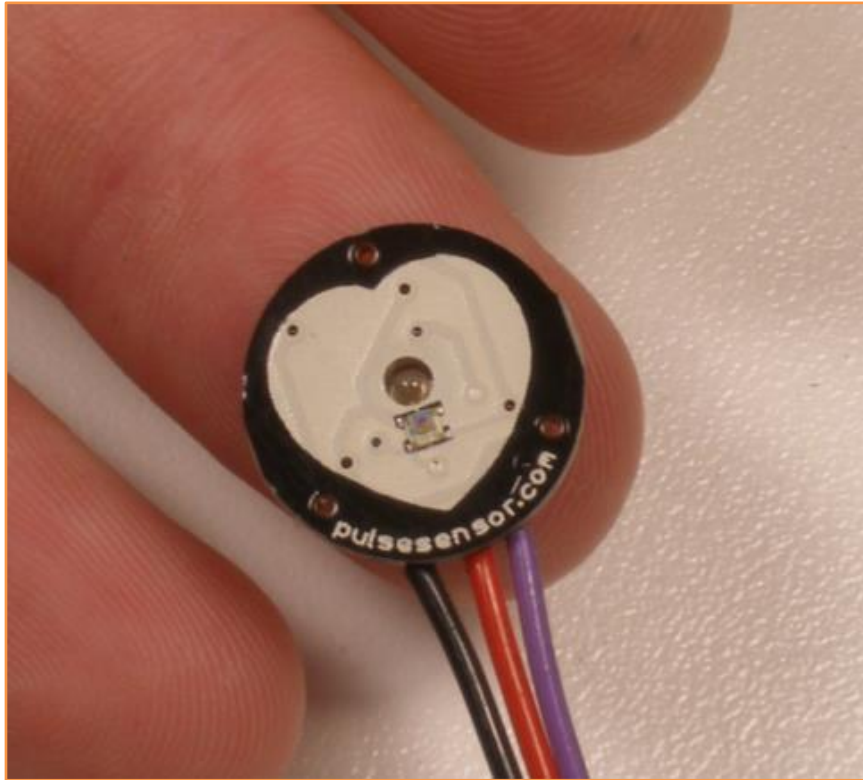


Fig. 1.5 : pulse sensor

The front of the sensor is the pretty side with the Heart logo. This is the side that makes contact with the skin. On the front you see a small round hole, which is where the LED shines through from the back, and there is also a little square just under the LED. The square is an ambient light sensor, exactly like the one used in cellphones, tablets, and laptops, to adjust the screen brightness in different light conditions. The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the light that bounces back.

The back of the sensor is where the rest of the parts are mounted. We put them there so they would not get in the way of the of the sensor on the front. Even the LED we are using is a reverse mount LED.

The Pulse Sensor has 3 holes around the outside edge which make it easy to sew it into almost anything.

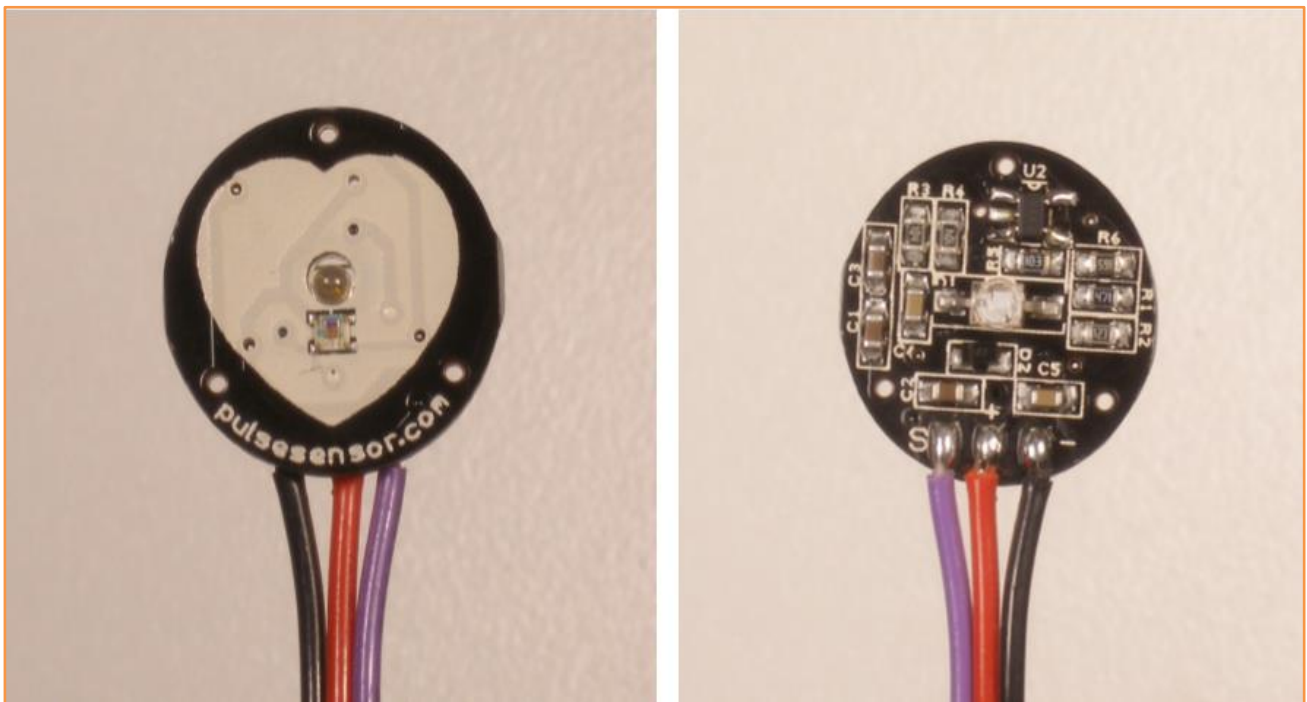


Fig. 1.6 : Front & behind shape of pules sensor

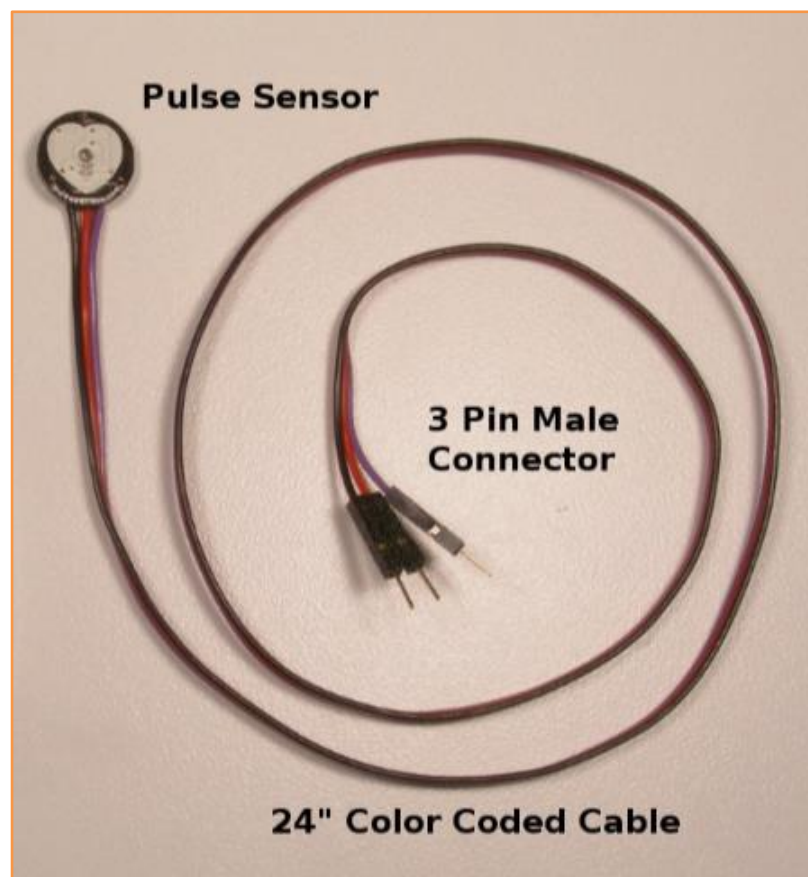


Fig. 1.7 : the full looking of sensor

A 24-inch Color-Coded Cable, with (male) header connectors. You'll find this makes it easy to embed the sensor into your project, and connect to an Arduino. No soldering is required.

- RED wire = +3V to +5V
- BLACK wire = GND
- PURPLE wire = Signal

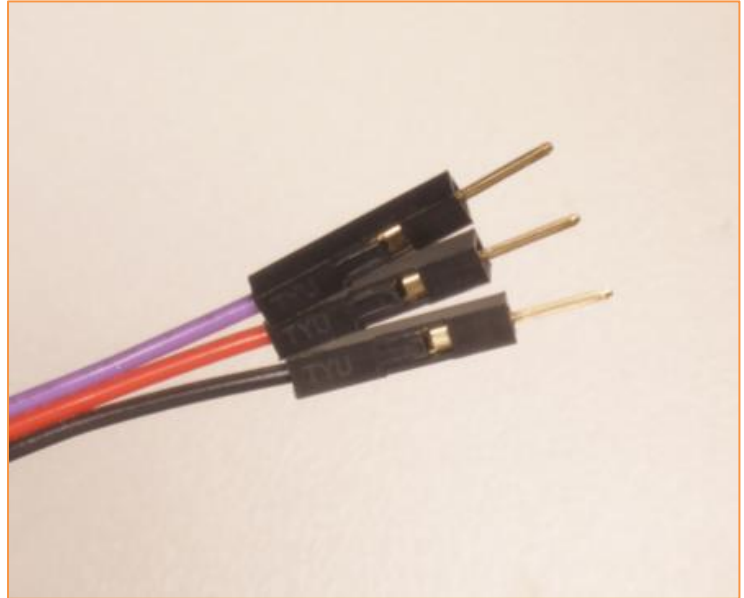


Fig. 1.8 : header connectors of pulse sensor

The Pulse Sensor can be connected to arduino, or plugged into a breadboard. Before we get it up and running, we need to protect the exposed circuitry so you can get a reliable heart beat signal.

1.2.3 : LCD I2C

A liquid crystal display is a special thin flat panel that can let light go through it, or can block the light . The panel is made up of several blocks, and each block can be in any shape. Each block is filled with liquid crystals that can be made clear or solid, by changing the electric current to that block. Liquid crystal displays are often abbreviated LCDs .

2 lines x 16 characters LCD display with WHITE characters on BLUE background and backlight.

Features :

- Wide viewing angle and high contrast
- Don't need separate power supply for backlight
- Supported 4 or 8 bit parallel interface
- Display 2-line X 16-character
- Operate with 5V DC
- Free 16 positions male header without I2C .
- Free 4 male header with I2C

Specifications :

- Module Size (W x H x T): 80mm X 36mm X 11mm
- Black Metal Bezel (W x H): 70.7mm X 23.8mm
- Viewing Area (W x H): 55.7mm X 11mm

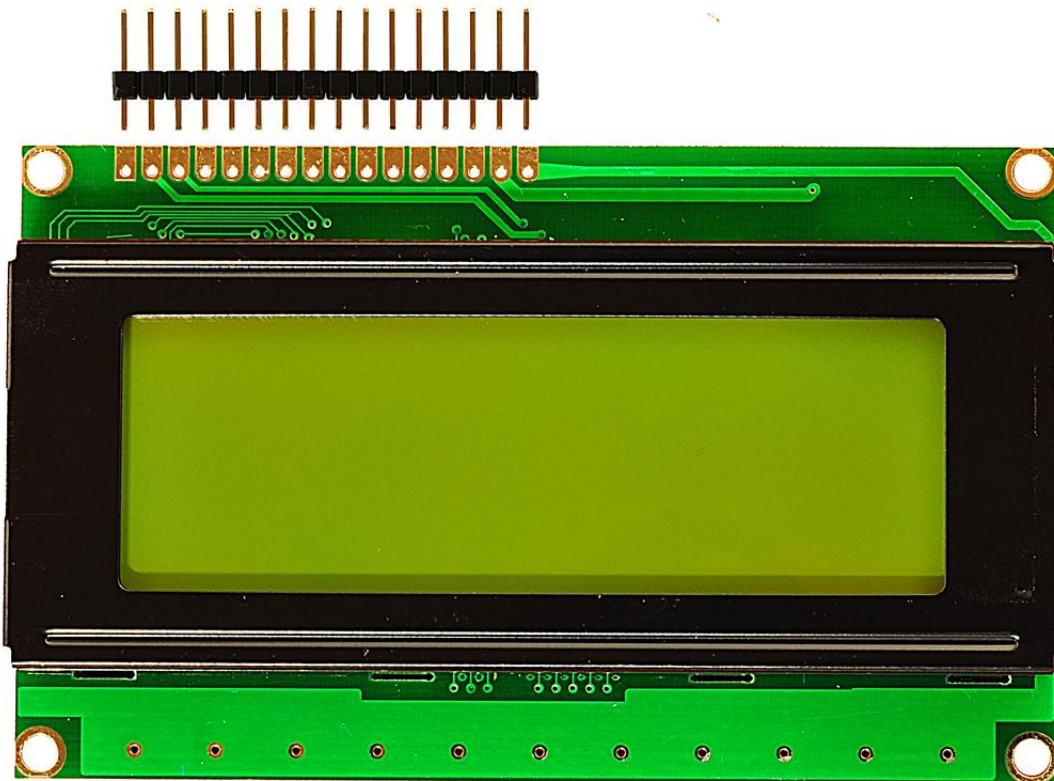


Fig. 1.9 : LCD 16*2

1.2.3.1 : LCD 16*2 Connected

LCD pins is modified to be 4 pins only after connected I2C module to LCD , the fig. below is shown the only way to the connection between I2C module and Arduino uno

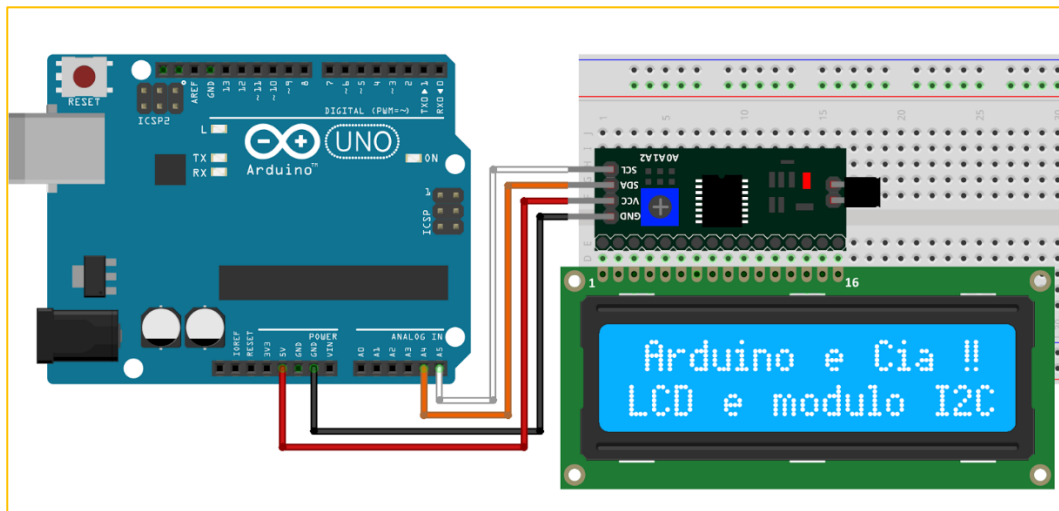


Fig. 1.10 : LCD I2C Connection with Arduino

First of all we connect i2c pins module as shown in the schematic. Power the LCD module to 5 volts and connect the ground as well. The SDA pin of the i2c module connected to arduino A5 and the SCL pin to A4. We connect the arduino to USB and we are ready to program.

1.2.3.2 : Source Coding

First thing we need to do is it insert the Liquid Crystal Library. We can do that like this: Sketch > Include Library > Wire.h > & > LiquidCrystal_I2C.h . Then we have to create an LCD object. In the setup we have to initialize the interface to the LCD and specify the dimensions of the display using the begin() function.

In the loop we write our main program. Using the `print()` function we print on the LCD. The `setCursor()` function is used for setting the location at which subsequent text written to the LCD will be displayed. The `blink()` function is used for displaying a blinking cursor and the `noBlink()` function for turning off. The `cursor()` function is used for displaying underscore cursor and the `noCursor()` function for turning off. Using the `clear()` function we can clear the LCD screen .

1.2.4 : Jumper Wire

Jumper wires are used for making connections between items on your board and your Arduino's header pins.

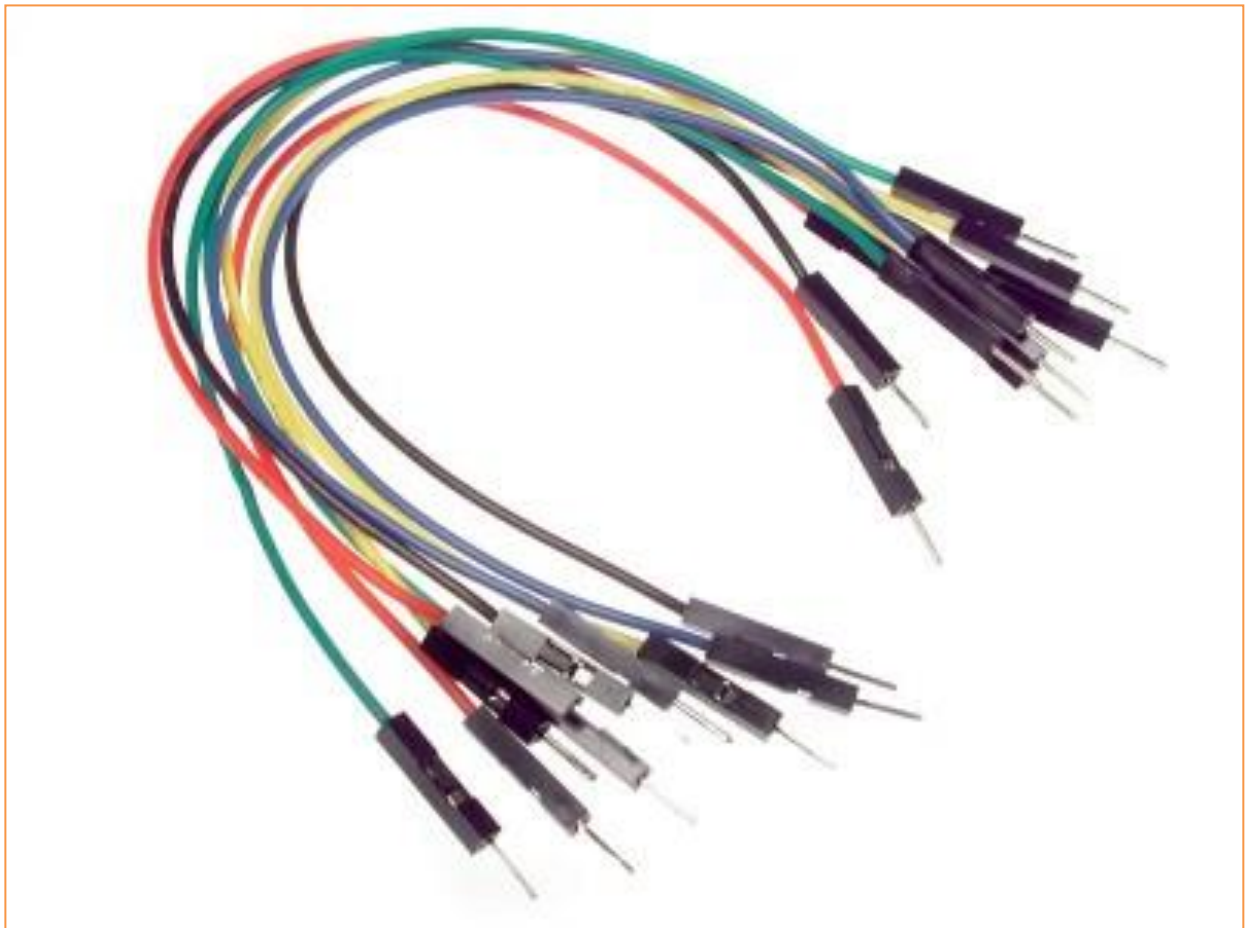


Fig. 1.11 : Jumper Wire

1.2.5 : Breadboard

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.

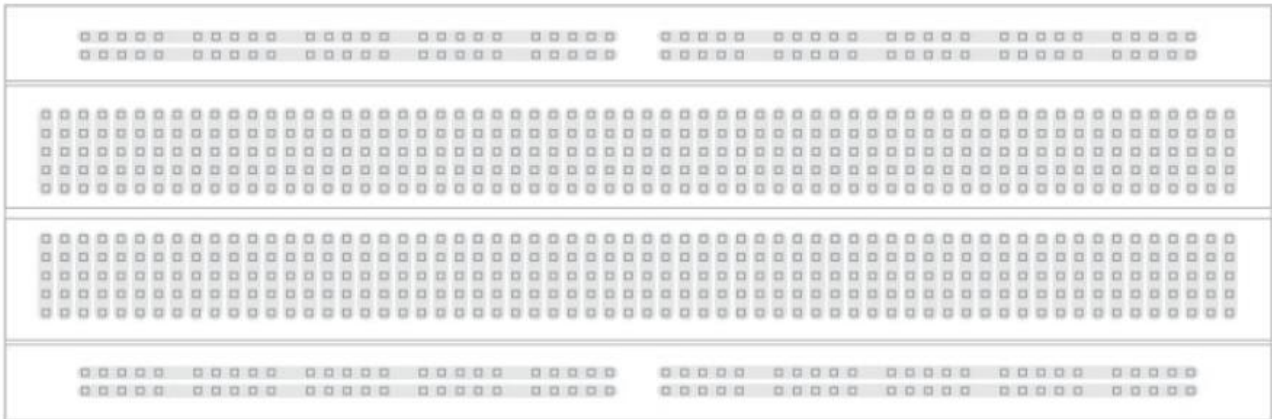


Fig. 1.12 : Breadboard

2.1 : The Circuits

2.1.1 : LCD Connection

We will Connect the LCD as shown in the fig. 2.1

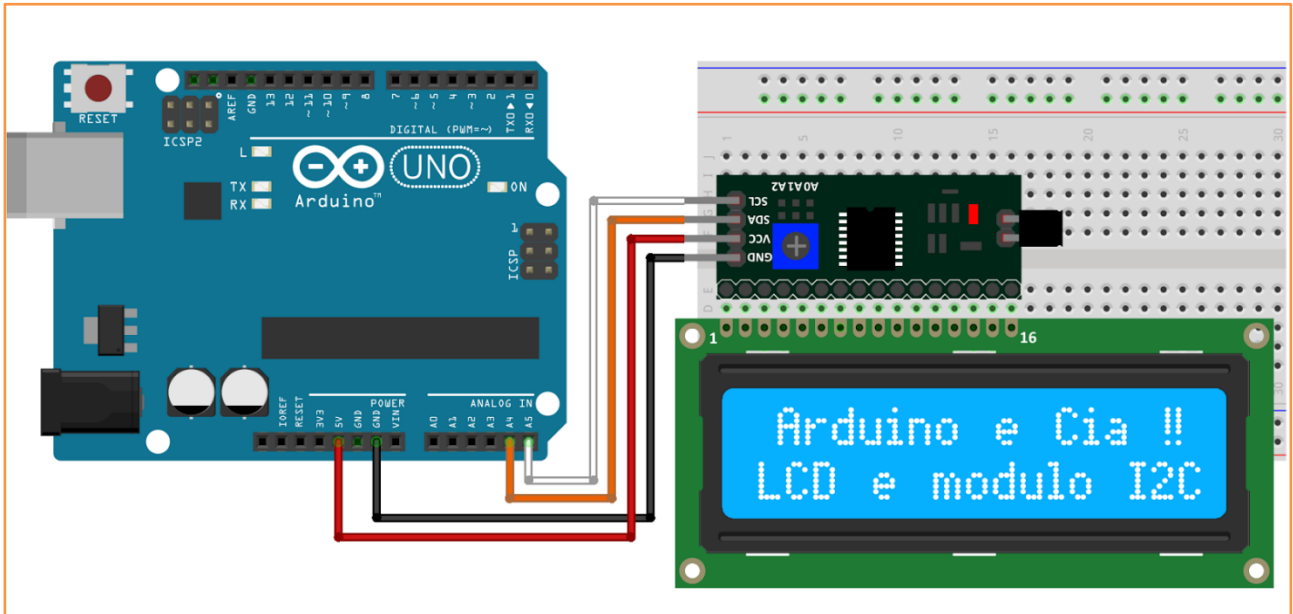


Fig. 2.1 : Connection of LCD with Arduino

2.1.2 : Pules Sensor Connection

We will Connect the pulse sensor as shown in the fig. 2.2

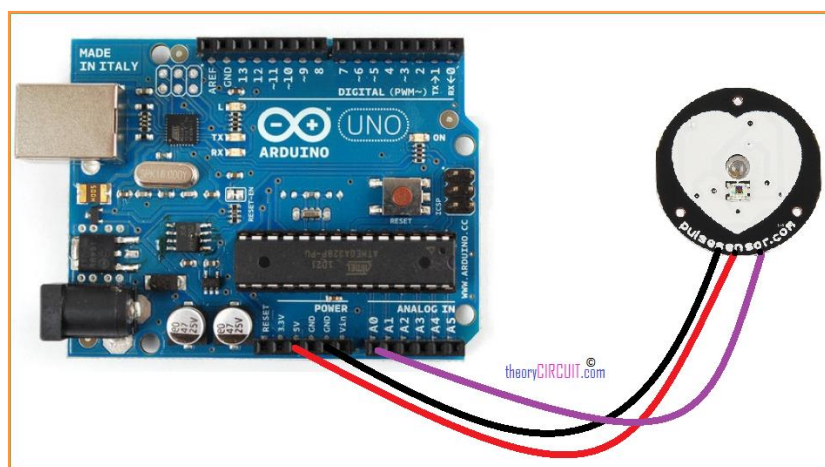


Fig. 2.2 : Connection of pulse sensor with Arduino

2.2 : Arduino Programming

After completing the connections of a circuit, we will program the Arduino with the Code

- First , must download the ARDUINO IDE program from the official website
(<https://www.arduino.cc/en/main/software>) .
- After download , we must to install it like any program .
- After the installing is done , the main screen of the program is shown in the figure 2.3

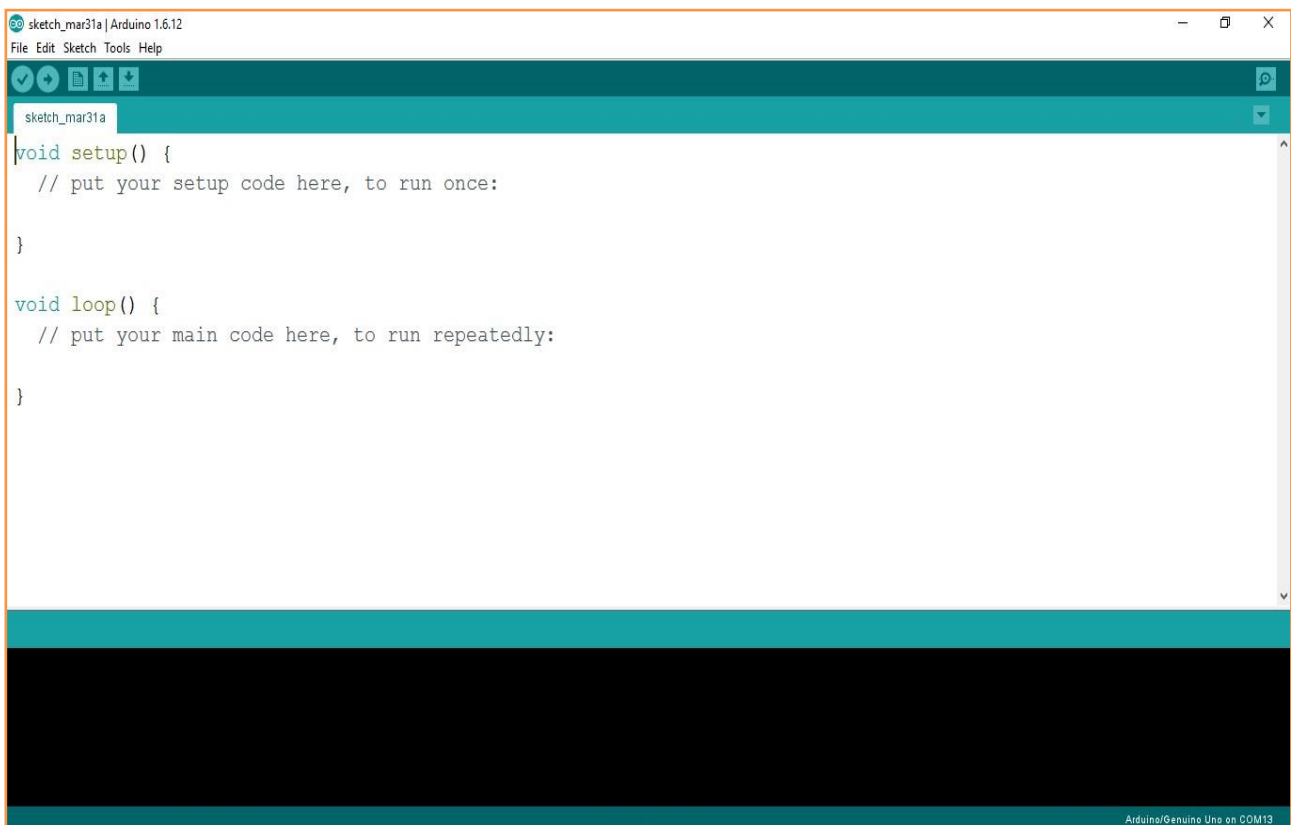


Fig. 2.3 : Arduino IDE

- Write the following code on the Arduino IDE program .

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,16,2); // set the LCD address to 0x27 for a 16 chars and
2 line display

int pulsePin = A0;           // Pulse Sensor purple wire connected to analog pin A0
int blinkPin = 13;          // pin to blink led at each beat

// Volatile Variables, used in the interrupt service routine!
volatile int BPM;            // int that holds raw Analog in 0. updated every 2mS
volatile int Signal;         // holds the incoming raw data
volatile int IBI = 600;      // int that holds the time interval between beats! Must
be seeded!
volatile boolean Pulse = false; // "True" when User's live heartbeat is detected.
"False" when not a "live beat".
volatile boolean QS = false; // becomes true when Arduino finds a beat.

static boolean serialVisual = true; // Set to 'false' by Default. Re-set to 'true' to see
Arduino Serial Monitor ASCII Visual Pulse

volatile int rate[10];        // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0; // used to determine pulse timing
volatile unsigned long lastBeatTime = 0; // used to find IBI
volatile int P = 512;         // used to find peak in pulse wave, seeded
volatile int T = 512;         // used to find trough in pulse wave, seeded
volatile int thresh = 525;     // used to find instant moment of heart beat, seeded
volatile int amp = 100;        // used to hold amplitude of pulse waveform,
seeded
volatile boolean firstBeat = true; // used to seed rate array so we startup with
reasonable BPM
volatile boolean secondBeat = false; // used to seed rate array so we startup with
reasonable BPM
```

```
void setup()
{
  pinMode(blinkPin,OUTPUT);    // pin that will blink to your heartbeat!
  Serial.begin(115200);        // we agree to talk fast!
  interruptSetup();            // sets up to read Pulse Sensor signal every 2mS
                                // IF YOU ARE POWERING The Pulse Sensor AT
                                // VOLTAGE LESS THAN THE BOARD VOLTAGE,
                                // UN-COMMENT THE NEXT LINE AND APPLY THAT
                                // VOLTAGE TO THE A-REF PIN
                                //  analogReference(EXTERNAL);

  lcd.init();
  lcd.init();
  lcd.backlight();
}

// Where the Magic Happens
void loop()
{
  serialOutput();

  if (QS == true) // A Heartbeat Was Found
  {
    // BPM and IBI have been Determined
    // Quantified Self "QS" true when arduino finds a heartbeat
    serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.
    QS = false; // reset the Quantified Self flag for next time
  }

  delay(20); // take a break
}

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 Found "); //ASCII Art Madness
    Serial.print("BPM: ");
    Serial.println(BPM);
    lcd.print("Heart-Beat Found ");
    lcd.setCursor(1,1);
    lcd.print("BPM: ");
    lcd.setCursor(5,1);
    lcd.print(BPM);
    delay(300);
    lcd.clear();
  }
}
```

```
}
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:
}

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 realistic 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
}
}

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 you're done!
} // end sir
```

- Connect the USB cable with the Arduino UNO board and Upload the code for it , we can upload the code from () or by using the keyboard (Ctrl + U) .