BIRZEIT UNIVERSITY

Faculty of Information Technology
Computer Systems Engineering Department

ENCS412, INTERFACING LABORATORY

EXP #1&2 .Report

Introduction to Arduino

&

Two Wire Interface I2C Bus
Multiplexed displays on Arduino

Name : Ra'fat Ahmad

ID: 1120521

Section#: 1

Dr. Wasel Ghanem

TA. Eng. Mohammed Mudalal

Date:18/10/2016

• Abstract

In this report I illustrate what is Arduino and it’s type then I discus I2C Bus then finally, i show you multiplexed displays on Arduino.

• Introduction & Theory

1. Introduction

In this report we will illustrate the Arduino microcontroller and the interfacing techniques used to use different types of sensors (LDR , 7 segment display , LM75B temperature sensor) , also the I2C bus connections and Software and Hardware Interrupts.

2. Theory

Arduino:

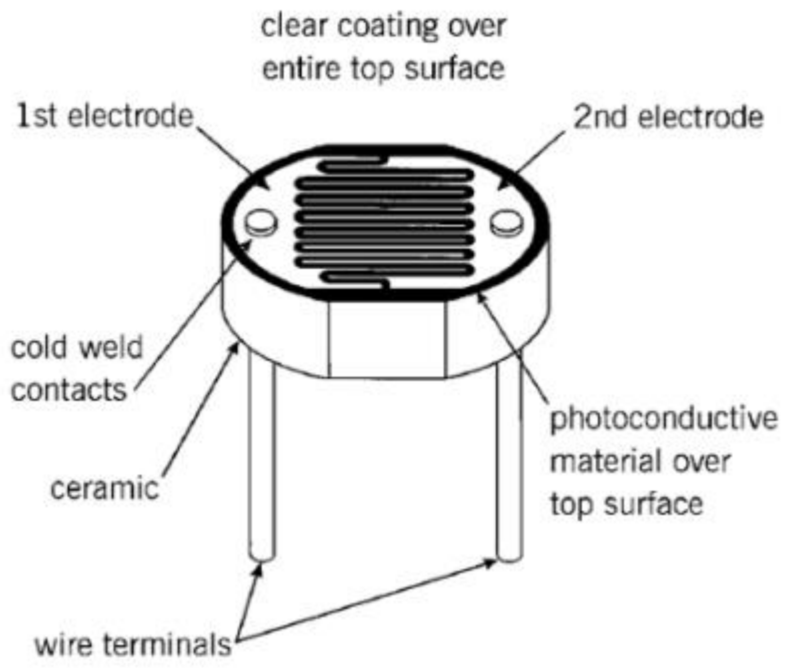
Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Photocells:

Photoresists, LDR (light dependent resistor)

Photocell sensors are light-detecting devices that act as automatic switches to power electrically powered devices on or off in the presence of light. One common use is in automatic lights that turn on at dusk and off at sunrise.


clear coating over entire top surface
1st electrode
2nd electrode
cold weld contacts
ceramic
photoconductive material over top surface
wire terminals

Interrupts:

An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. There are two types of interrupts: hardware interrupts and software interrupts.
Hardware interrupts:
A hardware interrupt is an electronic alerting signal sent to the processor from an external device. Arduino uno board has two external interrupts: int.0 (on

digital pin 2) and int.1 (on digital pin 3). when the triggering event is captured at these pins, the corresponding interrupt service routine is executed. The attachInterrupt() API is used to specify a function to call when an external interrupt occurs.

Software interrupts:
A software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. In this experiment we will focus on the interrupt caused by timer1 overflow.Timer1 is 16 bit hardware timer on the ATmega168/328. There are 3 hardware timers available on the chip, and they can be configured in a variety of ways to achieve different functionality. The timer can be programmed by some special registers. You can configure the prescaler for the timer, or the mode of operation and many other things. Timer1's clock speed is defined by setting the prescaler, or divisor. This prescale can be set to 1,8, 64, 256 or 1024. Timer Register You can change the Timer behaviour through the timer register

I2C:

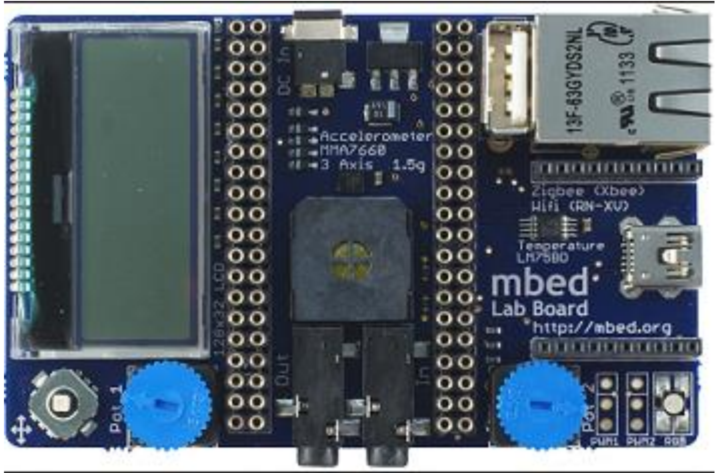
I^{2}C (Inter-Integrated Circuit) is a multi master serial single-ended computer bus ,used for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other digital electronic devices.

The I2C (Inter-IC) bus is a bi-directional two-wire serial bus that provides a communication link between integrated circuits (ICs). Phillips introduced the I2C bus 20 years ago for mass-produced items such as televisions, VCRs, and audio equipment. Today, I2C is the de-facto solution for embedded applications.

There are three data transfer speeds for the I2C bus: standard, fast-mode, and high-speed mode. Standard is 100 Kbps. Fast-mode is 400 Kbps, and high-speed mode supports speeds up to 3.4 Mbps. All are backward compatible. The I2C bus supports 7-bit and 10-bit address space devices and devices that operate under different voltages.

LM75B temperature sensor:

The LM75B is an I2C digital temperature sensor that is available in various packages. It has a range of -55^C to +125^C, with a 0.125C resolution.



The LM75B is a temperature-to-digital converter using an on-chip band gap temperature sensor and Sigma-Delta A-to-D conversion technique with an over temperature detection output. It can be communicated by a controller via the 2-wire serial I^{2}C-bus interface.

The temperature register always stores an 11-bit 2’s complement data giving a temperature resolution of 0.125 °C. This high temperature resolution is particularly useful in applications of measuring precisely the thermal drift or runaway. The LM75B powers up in the normal operation mode with the OS in comparator mode, temperature threshold of 80 °C and hysteresis of 75 °C, so that it can be used as a stand-alone thermostat with those pre-defined temperature set points.

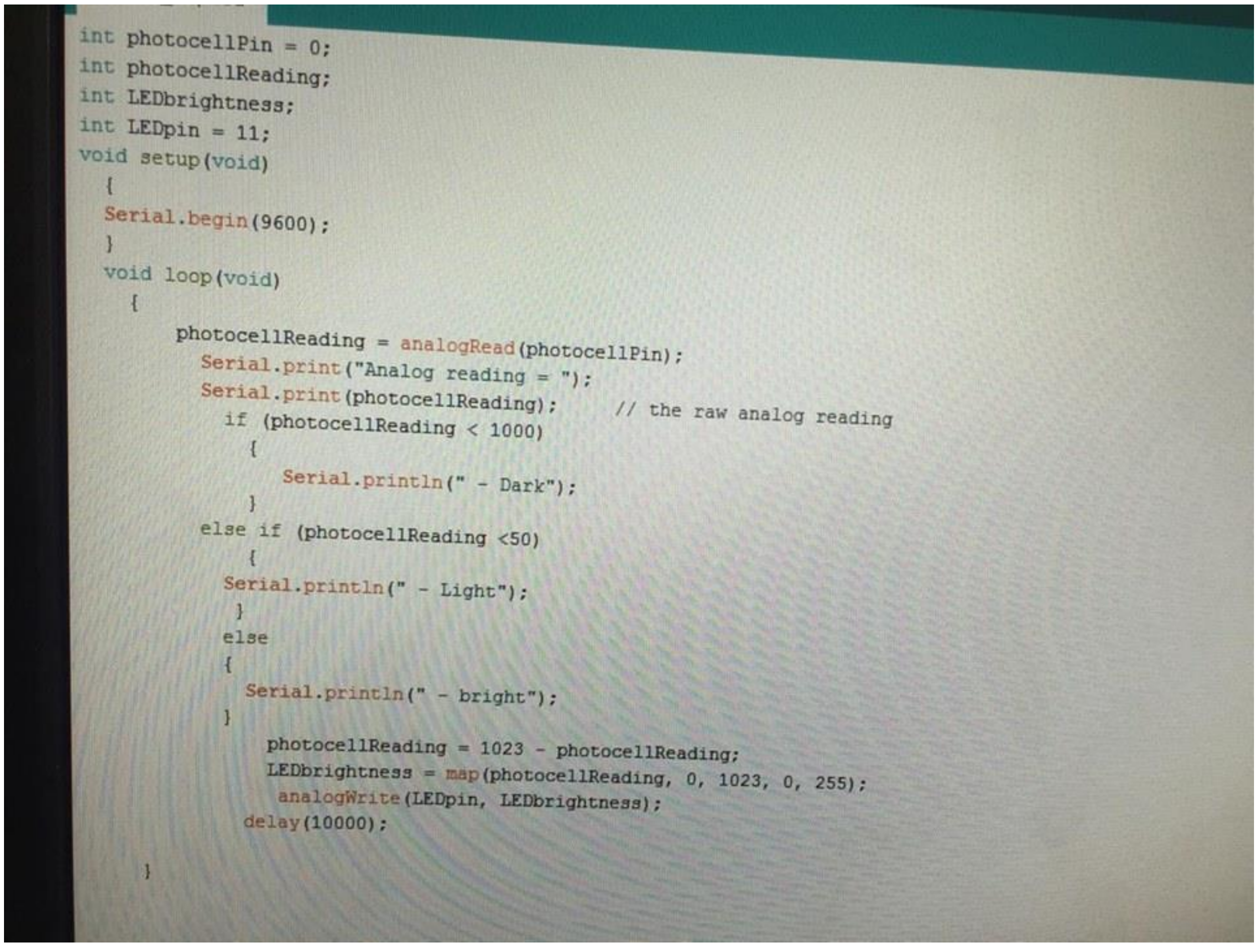
• Procedure and Discussion

After connecting the USB cable between the Arduino and the Lab's PC and installing the drivers for the Arduino on the Lab's PC, we can start compiling codes and uploading them to the Arduino after choosing the correct serial port, the code should have a SETUP and LOOP functions the SETUP function initializes and sets the initial values, the LOOP work infinitely as the code doesn't force it to stop.

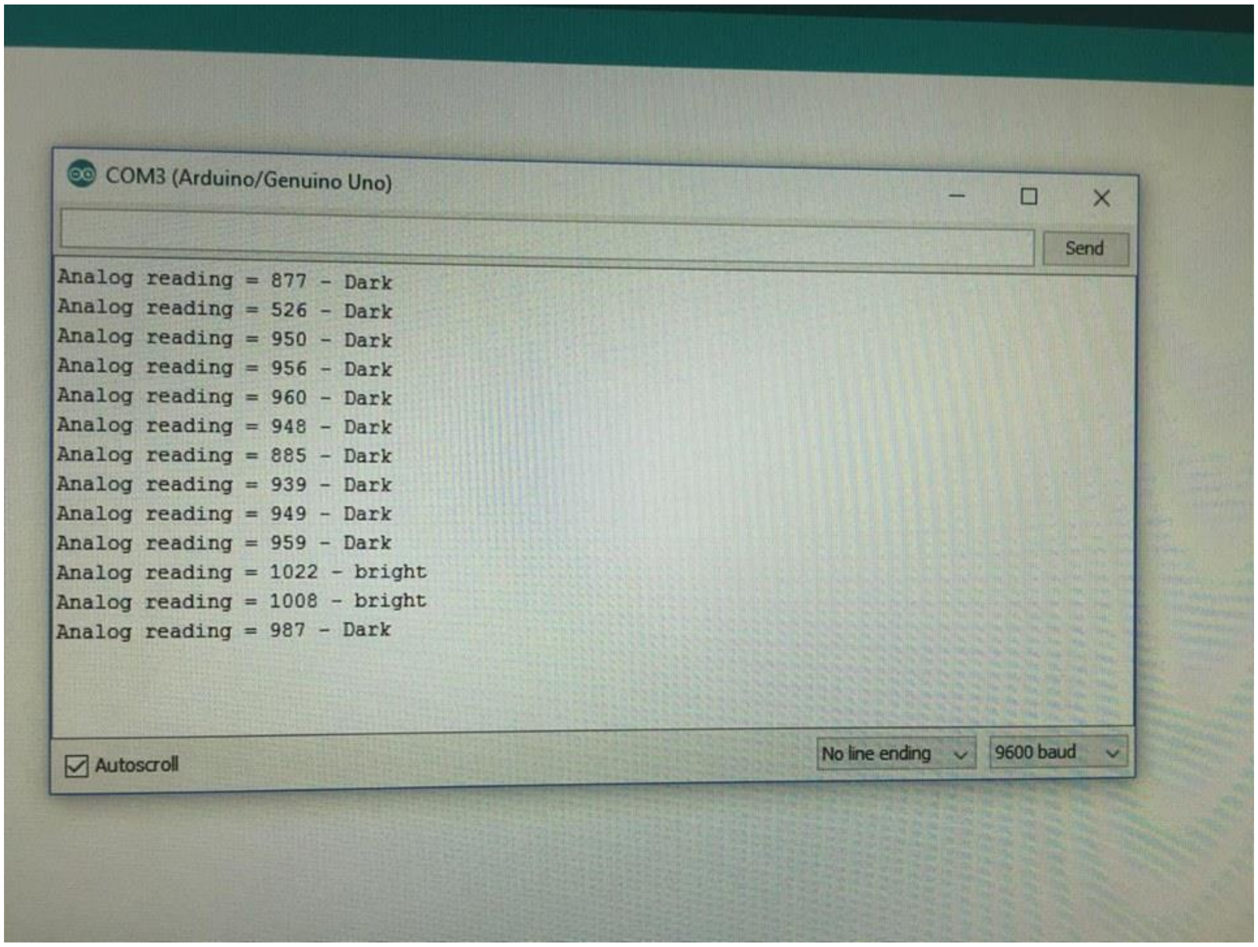
Photocells

In this part we will build a simple photocell test, after connecting the Photocell, Resistors, and a Led as in the figure bellow, the Led is connected to digital Pin11 and the read voltage from the photocell is read from analog Pin0, the function (analogRead) is used to read an analog value from the specified pin in the range of (0-1023), to determine when the led should be

ON or OFF as follow:



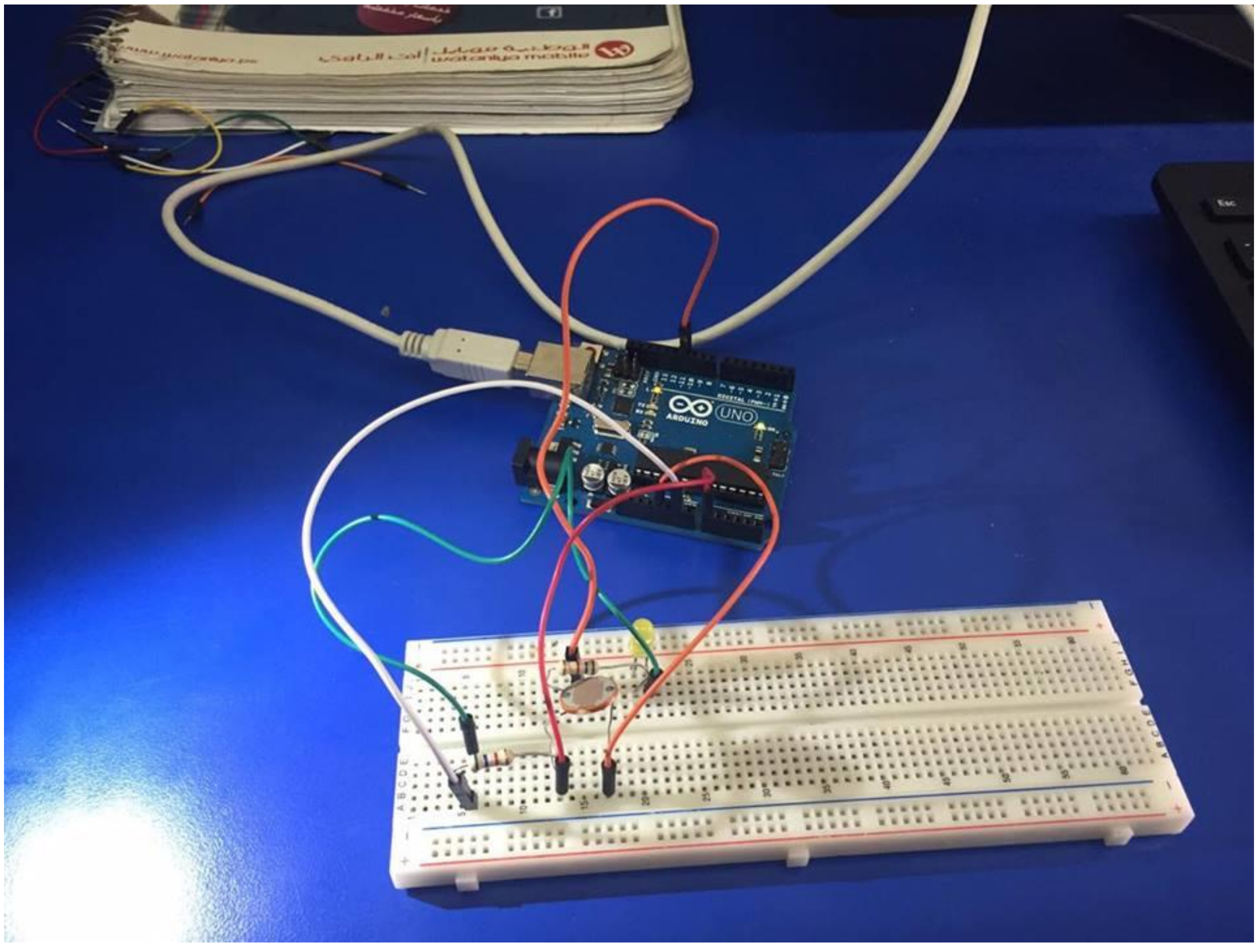
```
int photocellPin = 0;
int photocellReading;
int LEDbrightness;
int LEDpin = 11;
void setup(void)
  {
  Serial.begin(9600);
  }
  void loop(void)
    {
       photocellReading = analogRead(photocellPin);
         Serial.print("Analog reading = ");
         Serial.print(photocellReading);     // the raw analog reading
           if (photocellReading < 1000)
             {
                Serial.println(" - Dark");
             }
         else if (photocellReading <50)
             {
           Serial.println(" - Light");
            }
           else
           {
             Serial.println(" - bright");
           }
               photocellReading = 1023 - photocellReading;
               LEDbrightness = map(photocellReading, 0, 1023, 0, 255);
                analogWrite(LEDpin, LEDbrightness);
             delay(10000);

    }
```



```
Analog reading = 877 - Dark
Analog reading = 526 - Dark
Analog reading = 950 - Dark
Analog reading = 956 - Dark
Analog reading = 960 - Dark
Analog reading = 948 - Dark
Analog reading = 885 - Dark
Analog reading = 939 - Dark
Analog reading = 949 - Dark
Analog reading = 959 - Dark
Analog reading = 1022 - bright
Analog reading = 1008 - bright
Analog reading = 987 - Dark
```

☑ Autoscroll   No line ending   9600 baud

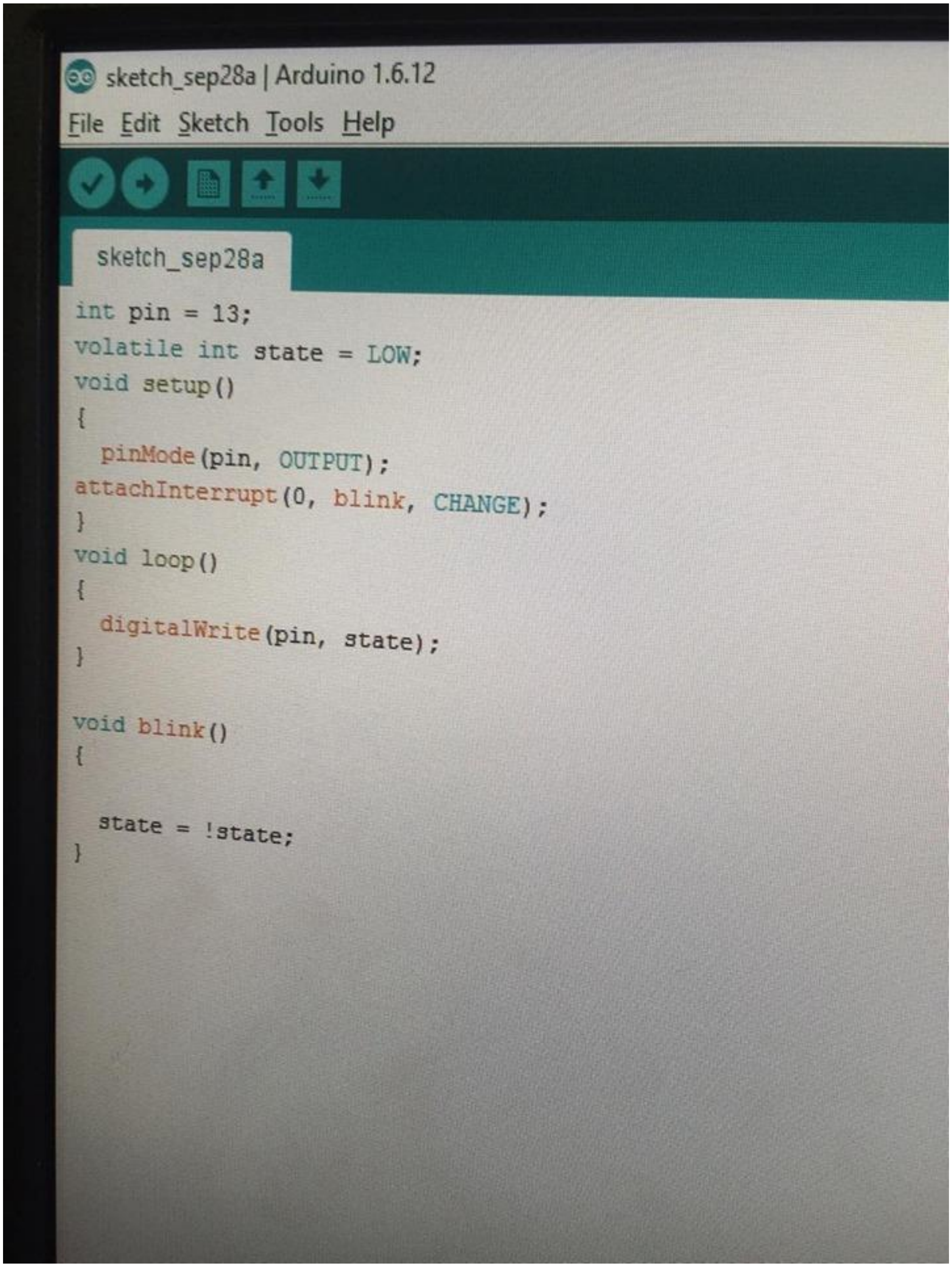
The connection of the circuit is shown below:



Hardware Interrupts

The (attachInterrupt) API is used to specify a function to call
when an external interrupt occurs as follow:
attachInterrupt(interrupt, function, mode)

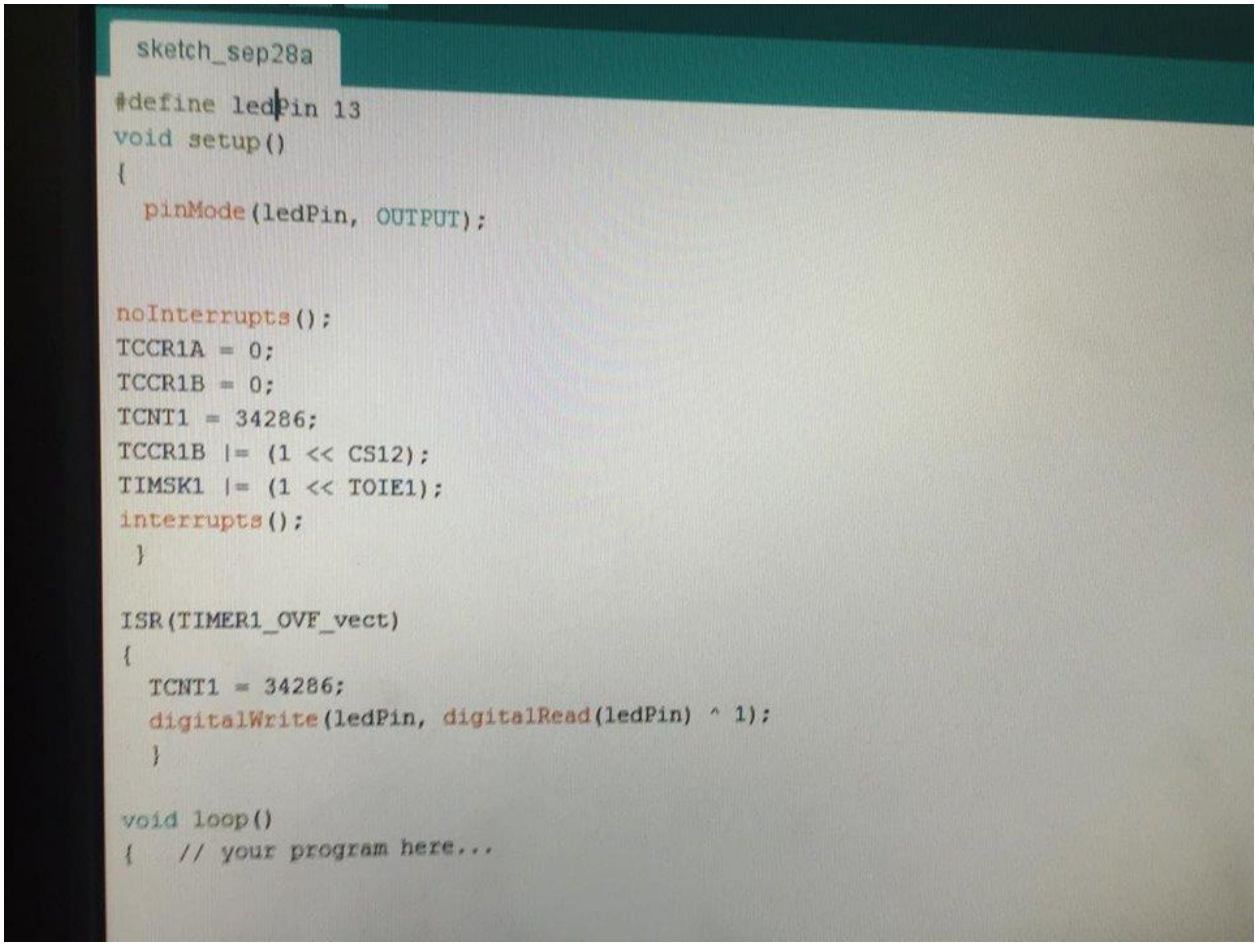
The code here is built to change the status of the Led on the Arduino
board which connected to Pin13 so that the hardware interrupt occurs when
a push button connected to push digital Pin2 is pushed, and then the function
(attachInterrupt) will call the function (blind) whenever
(change) occurs, since the push button is connected with a resistor and
5volt, whenever the push button pushed the value changes from 0 to 1 and
when it released it will change from 1-0 so the change occurs, the
(attachInterrupt) call the function (blind) which changes the Led
status from LOW to HIGH and vice versa.



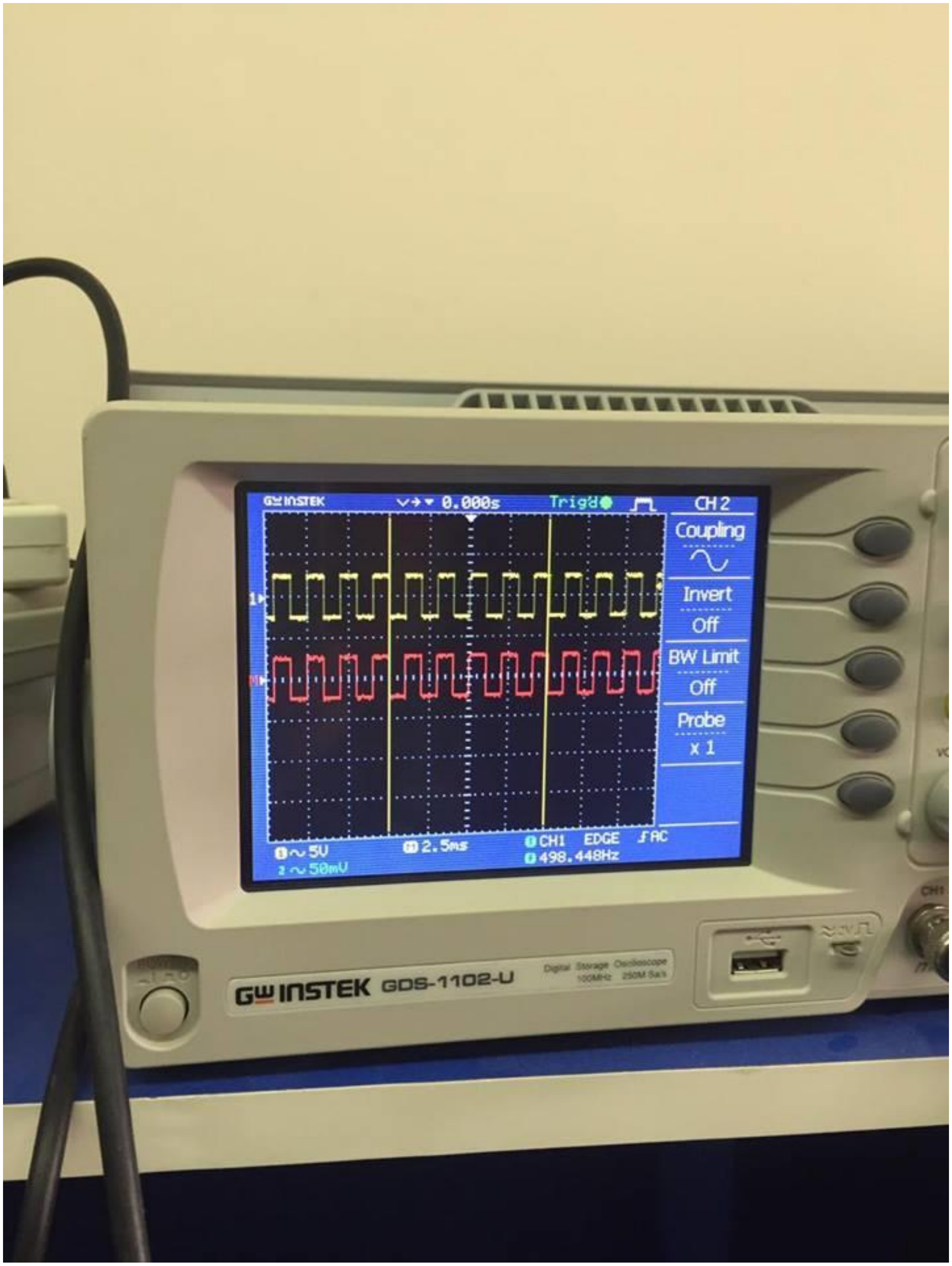
```
int pin = 13;
volatile int state = LOW;
void setup()
{
  pinMode(pin, OUTPUT);
attachInterrupt(0, blink, CHANGE);
}
void loop()
{
  digitalWrite(pin, state);
}

void blink()
{

  state = !state;
}
```

Software Interrupts

The code here uses a software interrupt using the timer (TCCR1B), the preload timer was calculated as follow: Preload = 65536 – (16Mhz / 256 / 2*1Hz) The software interrupt occurs when the buffer is full, so when the timer overflow it activates Software interrupt, and using a predefined prescaler as (256), the value of the (CS12) should be set in the register (TCCR1B), then we should enable the overflow interrupt (TOIE1), so the code is to turn on the Led every 0.5 second.The (To do) was to generate a square wave on Pin13 with freq 1Khz:
Preload = 65536 – (16Mhz / 256 / 2*1KHz)
= 31250 < 65536



```
#define ledPin 13
void setup()
{
  pinMode(ledPin, OUTPUT);


noInterrupts();
TCCR1A = 0;
TCCR1B = 0;
TCNT1 = 34286;
TCCR1B |= (1 << CS12);
TIMSK1 |= (1 << TOIE1);
interrupts();
 }

ISR(TIMER1_OVF_vect)
{
  TCNT1 = 34286;
  digitalWrite(ledPin, digitalRead(ledPin) ^ 1);
  }

void loop()
{   // your program here...
```



0.000s Trig'd

CH 2

Coupling ∼

Invert Off

BW Limit Off

Probe x 1

1 ∼ 5V

2 ∼ 50mV

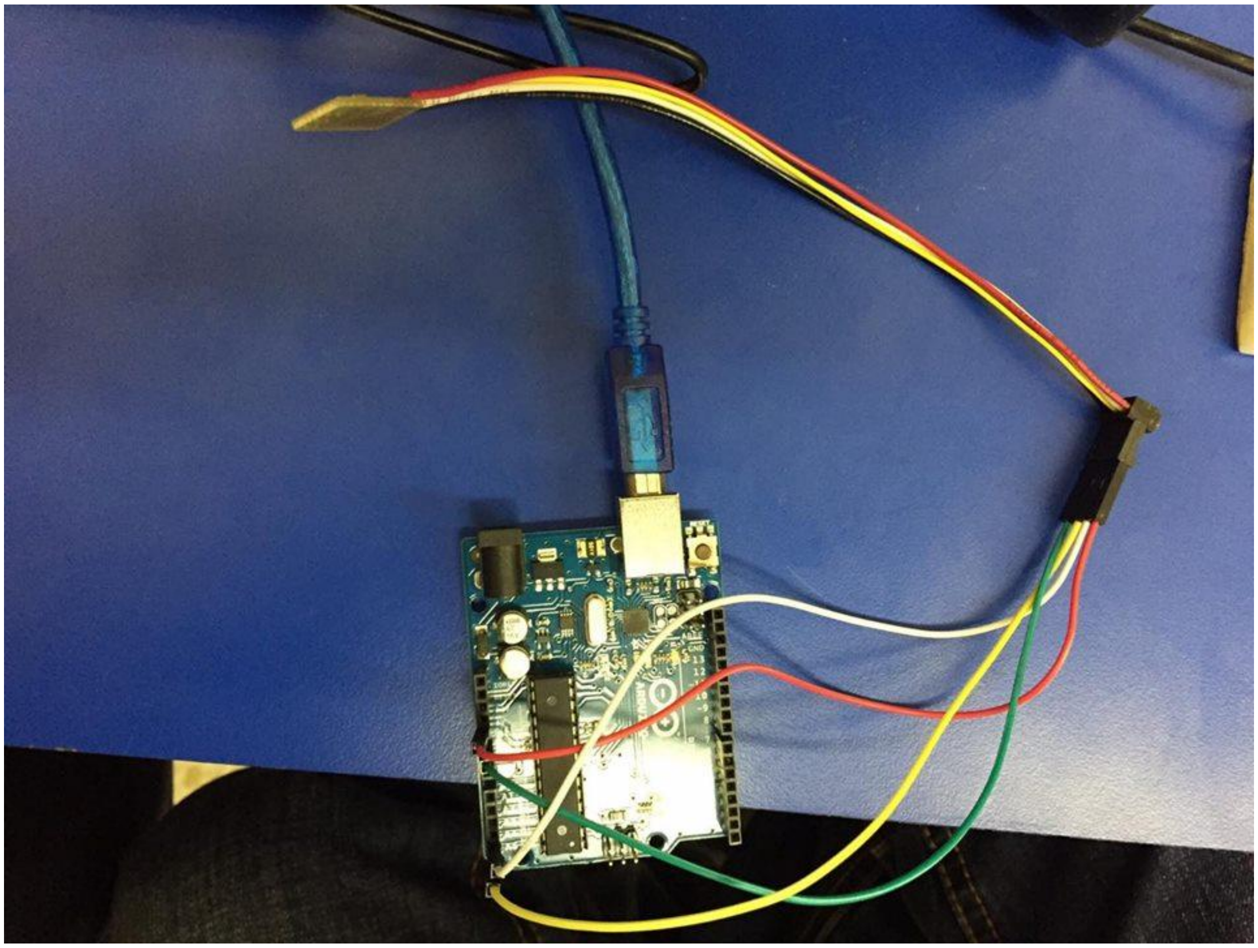
2.5ms

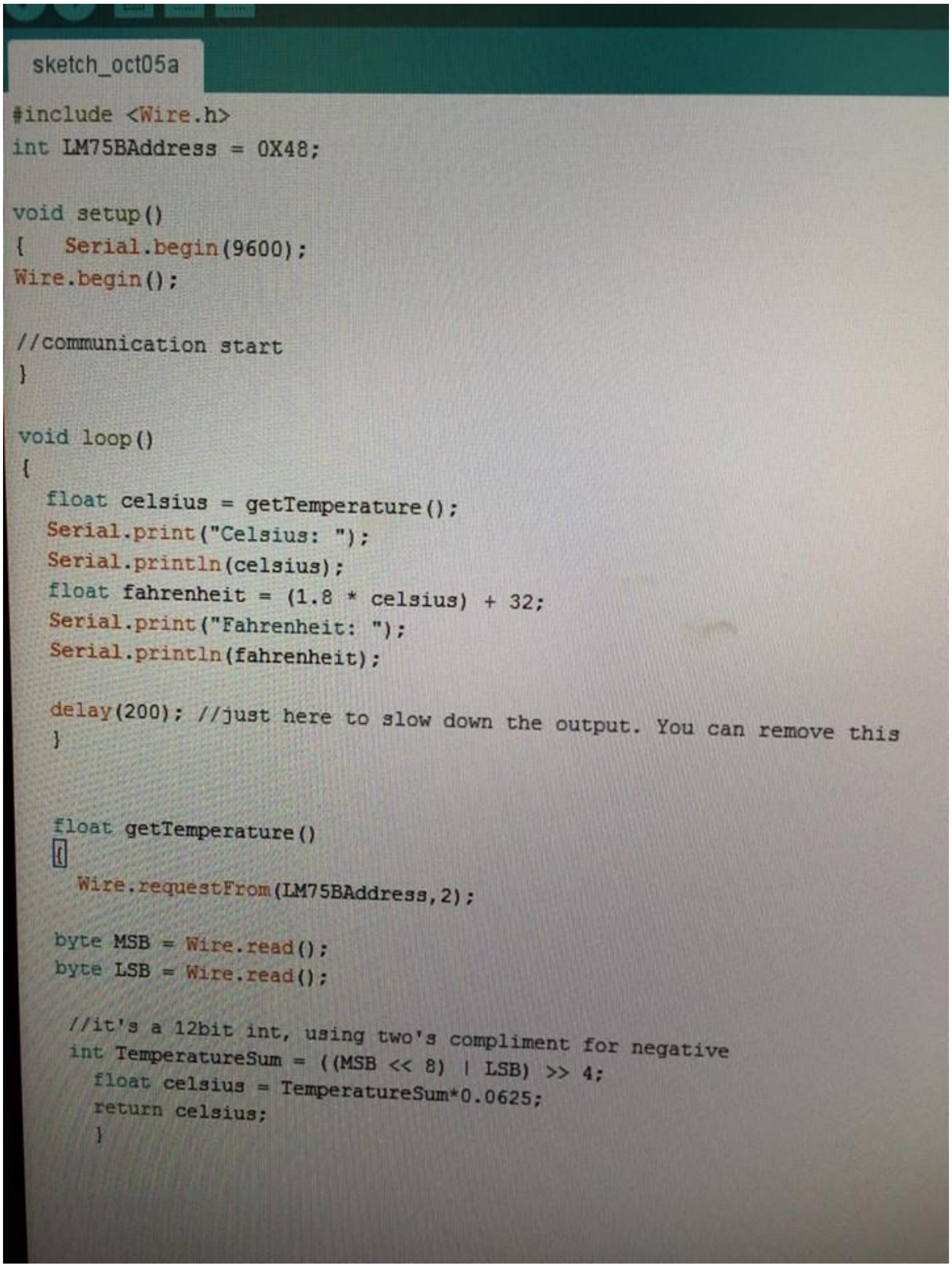
CH1 EDGE AC

498.448Hz

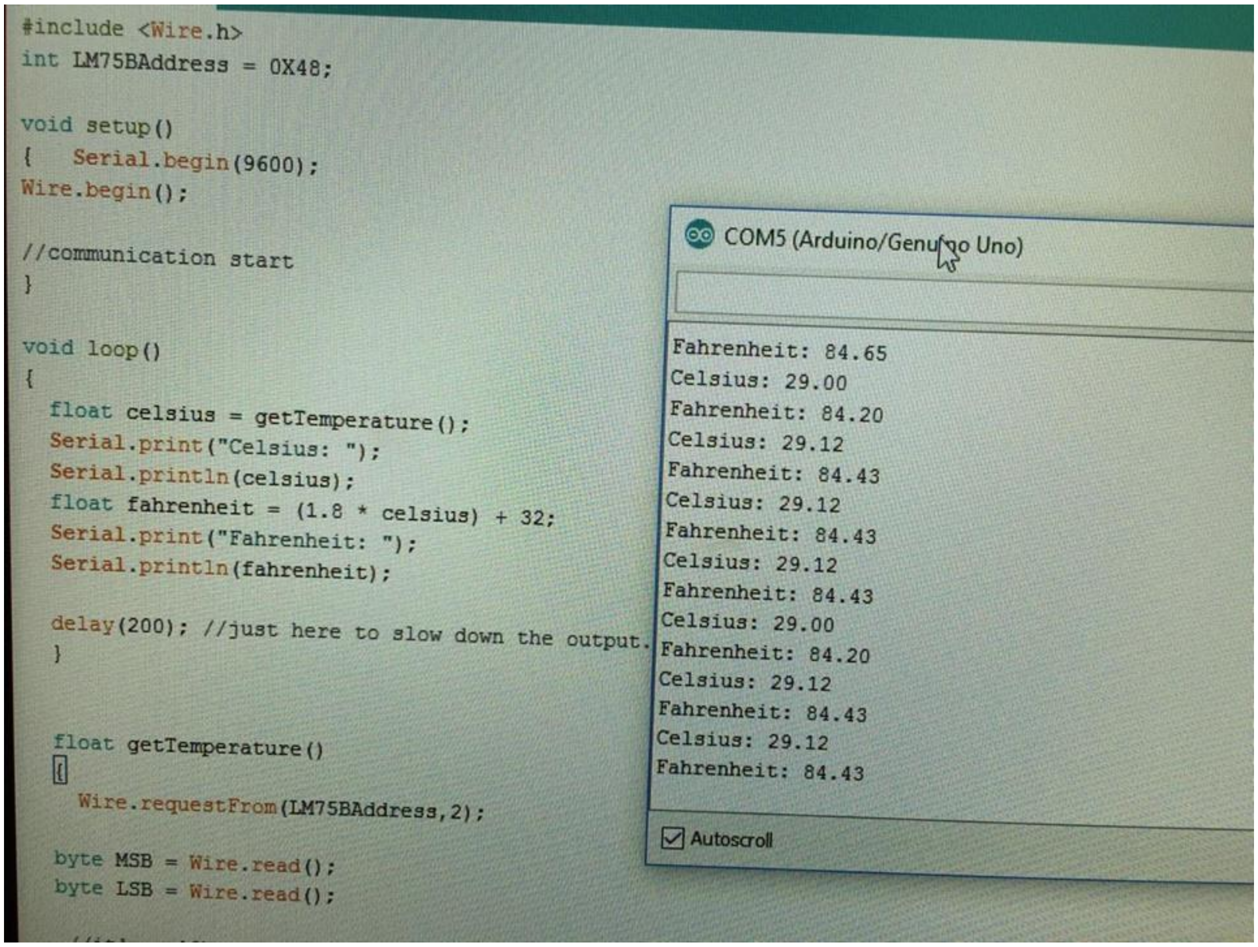
LM75B

Here we will use LM75B temperature sensor to measure the temperature and display the values on the (Serial Monitor), the address of the LM75B sensor is (1001000) since (A2, A1, A0) is grounded.




sketch_oct05a

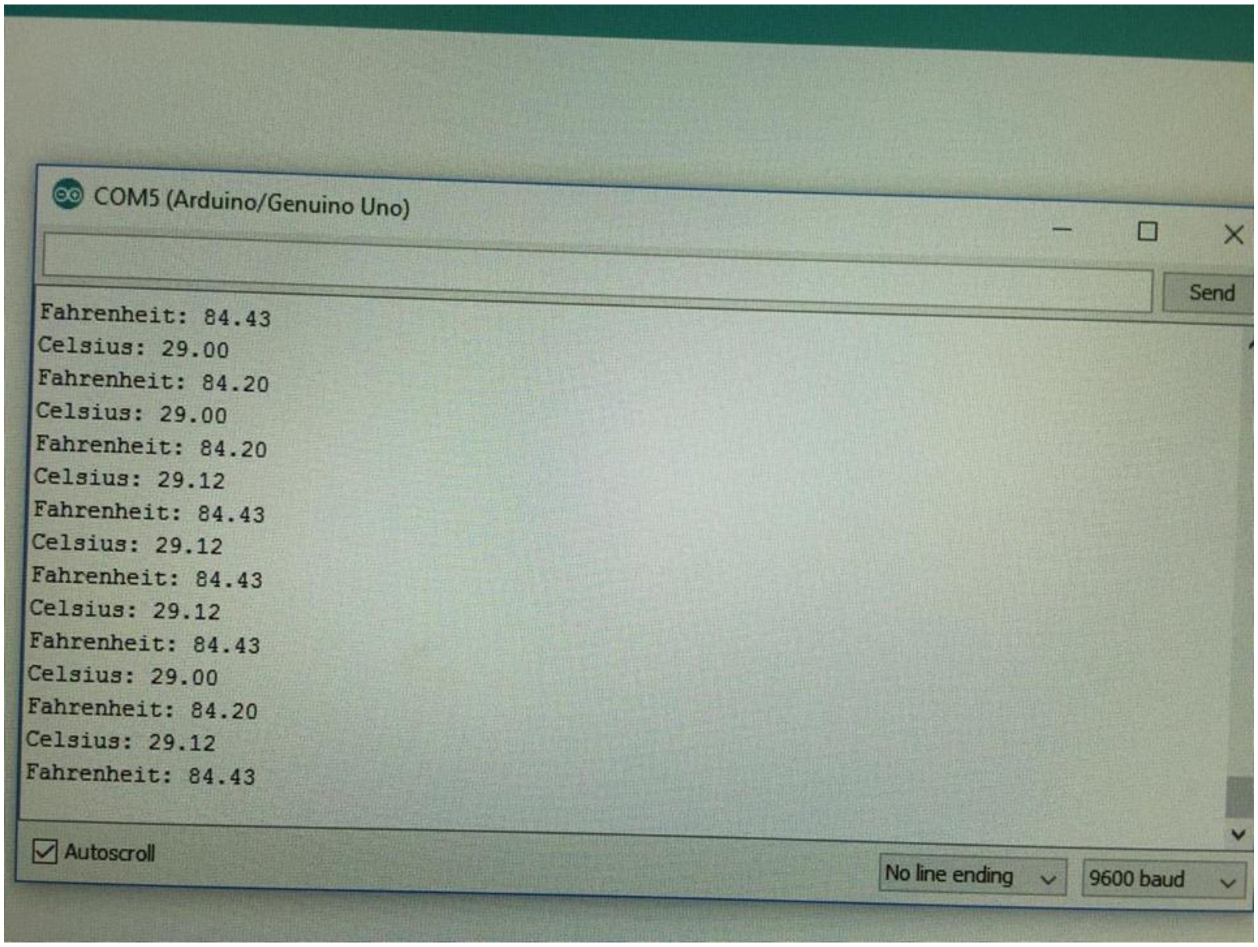
```
#include <Wire.h>
int LM75BAddress = 0X48;

void setup()
{   Serial.begin(9600);
Wire.begin();

//communication start
}

void loop()
{
  float celsius = getTemperature();
  Serial.print("Celsius: ");
  Serial.println(celsius);
  float fahrenheit = (1.8 * celsius) + 32;
  Serial.print("Fahrenheit: ");
  Serial.println(fahrenheit);

  delay(200); //just here to slow down the output. You can remove this
  }


  float getTemperature()
  {
    Wire.requestFrom(LM75BAddress,2);

  byte MSB = Wire.read();
  byte LSB = Wire.read();

   //it's a 12bit int, using two's compliment for negative
   int TemperatureSum = ((MSB << 8) | LSB) >> 4;
     float celsius = TemperatureSum*0.0625;
     return celsius;
     }
```



```
#include <Wire.h>
int LM75BAddress = 0X48;

void setup()
{   Serial.begin(9600);
Wire.begin();

//communication start
}

void loop()
{
  float celsius = getTemperature();
  Serial.print("Celsius: ");
  Serial.println(celsius);
  float fahrenheit = (1.8 * celsius) + 32;
  Serial.print("Fahrenheit: ");
  Serial.println(fahrenheit);

  delay(200); //just here to slow down the output.
  }


  float getTemperature()
  {
    Wire.requestFrom(LM75BAddress,2);

  byte MSB = Wire.read();
  byte LSB = Wire.read();
```

COM5 (Arduino/Genuino Uno)

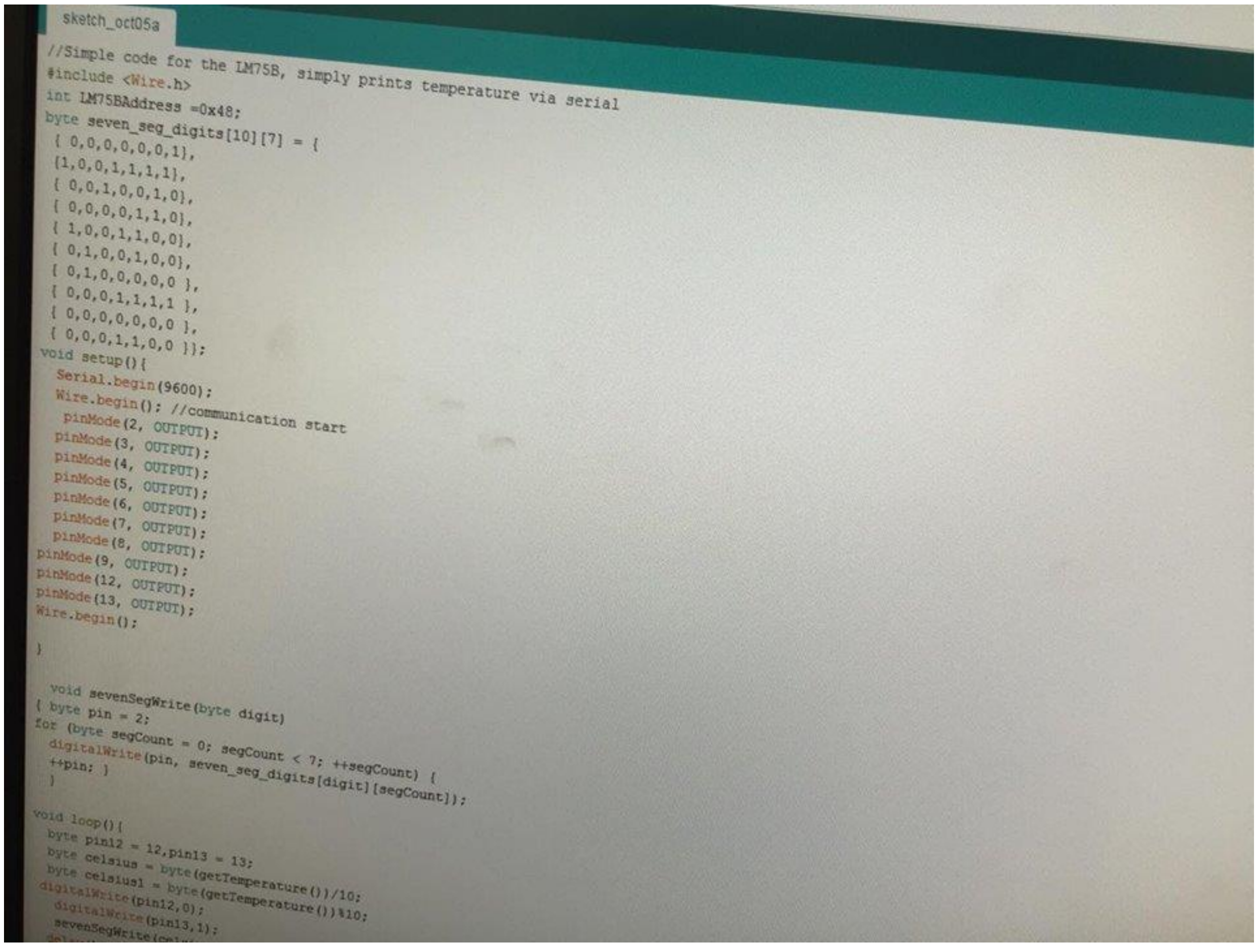
```
Fahrenheit: 84.65
Celsius: 29.00
Fahrenheit: 84.20
Celsius: 29.12
Fahrenheit: 84.43
Celsius: 29.12
Fahrenheit: 84.43
Celsius: 29.12
Fahrenheit: 84.43
Celsius: 29.00
Fahrenheit: 84.20
Celsius: 29.12
Fahrenheit: 84.43
Celsius: 29.12
Fahrenheit: 84.43
```



```
Fahrenheit: 84.43
Celsius: 29.00
Fahrenheit: 84.20
Celsius: 29.00
Fahrenheit: 84.20
Celsius: 29.12
Fahrenheit: 84.43
Celsius: 29.12
Fahrenheit: 84.43
Celsius: 29.12
Fahrenheit: 84.43
Celsius: 29.00
Fahrenheit: 84.20
Celsius: 29.12
Fahrenheit: 84.43
```

7 Segment Display

In this part we will use the 7Seg Display to display the numbers (0-9), the pins that we connect the Seven Segments Display are (2, 3, 4, 5, 6, 7, 8) and using the common anode display, we have complete the array for the rest of the numbers as follow :



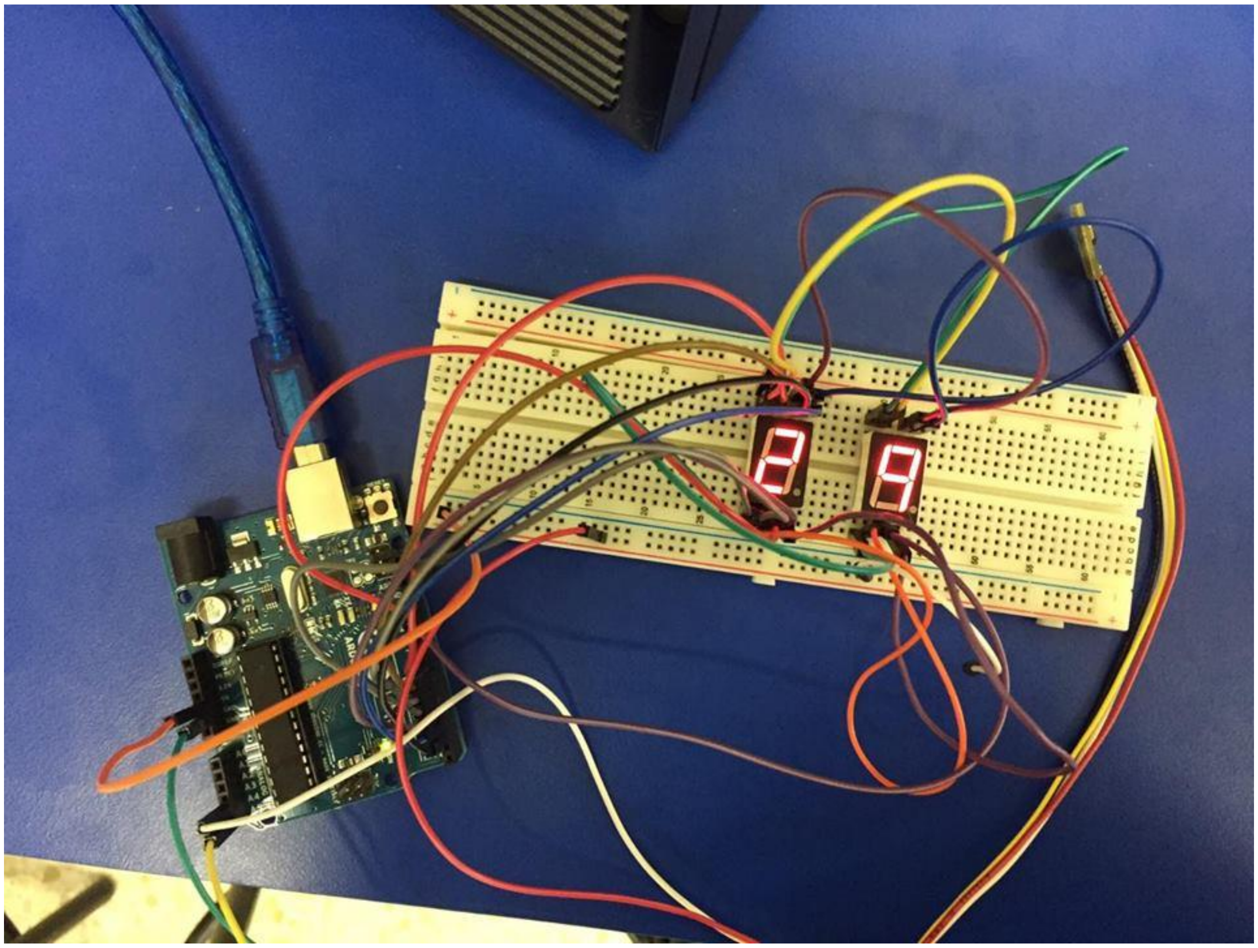
sketch_oct05a

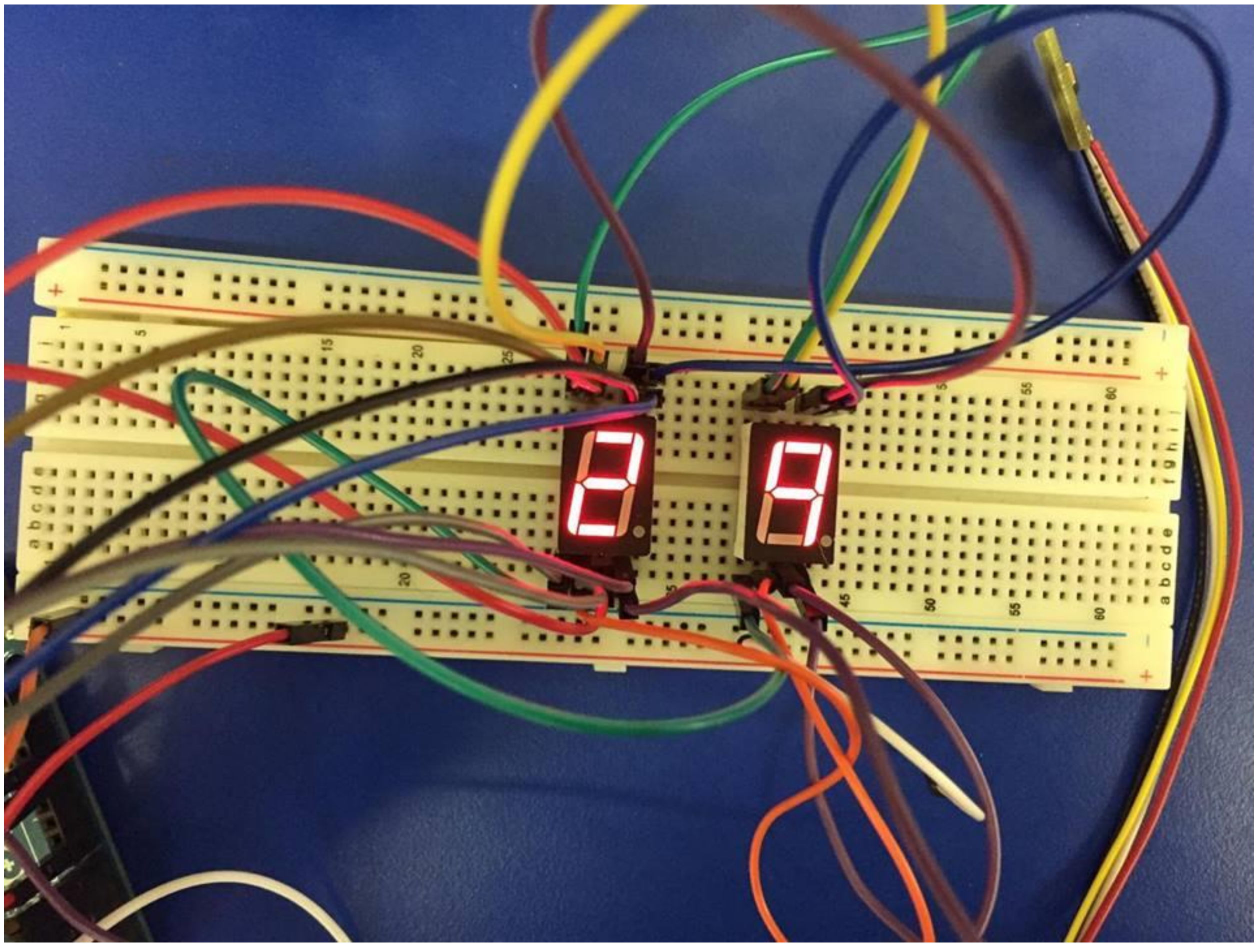
```
//Simple code for the LM75B, simply prints temperature via serial
#include <Wire.h>
int LM75BAddress =0x48;
byte seven_seg_digits[10][7] = {
 { 0,0,0,0,0,0,1},
 {1,0,0,1,1,1,1},
 { 0,0,1,0,0,1,0},
 { 0,0,0,0,1,1,0},
 { 1,0,0,1,1,0,0},
 { 0,1,0,0,1,0,0},
 { 0,1,0,0,0,0,0 },
 { 0,0,0,1,1,1,1 },
 { 0,0,0,0,0,0,0 },
 { 0,0,0,1,1,0,0 }};
void setup(){
  Serial.begin(9600);
  Wire.begin(); //communication start
   pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(12, OUTPUT);
pinMode(13, OUTPUT);
Wire.begin();

}


  void sevenSegWrite(byte digit)
{ byte pin = 2;
for (byte segCount = 0; segCount < 7; ++segCount) {
  digitalWrite(pin, seven_seg_digits[digit][segCount]);
  ++pin; }
  }

void loop(){
  byte pin12 = 12,pin13 = 13;
  byte celsius = byte(getTemperature())/10;
  byte celsius1 = byte(getTemperature())%10;
 digitalWrite(pin12,0);
  digitalWrite(pin13,1);
  sevenSegWrite(cel
```

LM75M && 7 Segment Display (To Do)

In this part we will use the LM75M temperature sensor and display the values using the (7 Segments Display) all the 2 digits, the problem was in the number of pins on the Arduino, we need 7 pins for each one, using the trick of time, we have displayed all the 2 digits each one on a (7 Segment Display) by connecting each 7 segment pin with the others pin, and using 2 other pins to control which one to enable, we set a very small time between each pulse so the values are displayed as 2 digits.

The following pictures show the results:





- ## Conclusion:

in the first part of these experiments we illustrate the Arduino environment and the specification of Arduino microcontroller and how we use it to interface simple sensors like LDR to Arduino to read analog or digital values and process it to do specific task , and in the second part we introduce the Hardware and software interrupts in Arduino , for software interrupt we manage to generate different clock speeds depending on the basic frequency in the Arduino which is very useful for various application that need to have accurate timing .

in the second part we deal with I2C bus communication using serial data line (SDA) & serial clock line (SCL) which is connected to temperature sensor to read the temperature values , so the data is processed on the Arduino then displayed on 7 segment display , and in the to do we handle multiple segment display at the same time by switching the enable so fast so we can minimize the number of pins used .

- **References:**

1. https://www.arduino.cc/en/Guide/Introduction
   Accessed on 20-10-2016 ,10:13 PM
2. https://www.reference.com/home-garden/photocell-sensor-771781909d9cce27#
   Accessed on 20-10-2016 ,10:33 PM
3. http://whatis.techtarget.com/definition/I2C-bus-Inter-IC-bus
   Accessed on 20-10-2016 ,10:56 PM
4. https://www.google.com/search?q=What%20is%20LM75B&newwindow=1&client=aff-maxthon-maxthon4&hs=6At&affdom=maxthon.com&channel=t23&source=lnms&tbm=isch&sa=X&ei=y_gdVcmZFdDSoASy7oC4CA&ved=0CAgQ_AUoAQ#imgrc=e6uJAVABQSvigM%3A
   Accessed on 20-10-2016 ,11:09 PM

5. http://www.dictionary.com/browse/photocell
Accessed on 20-10-2016 ,11:33 PM