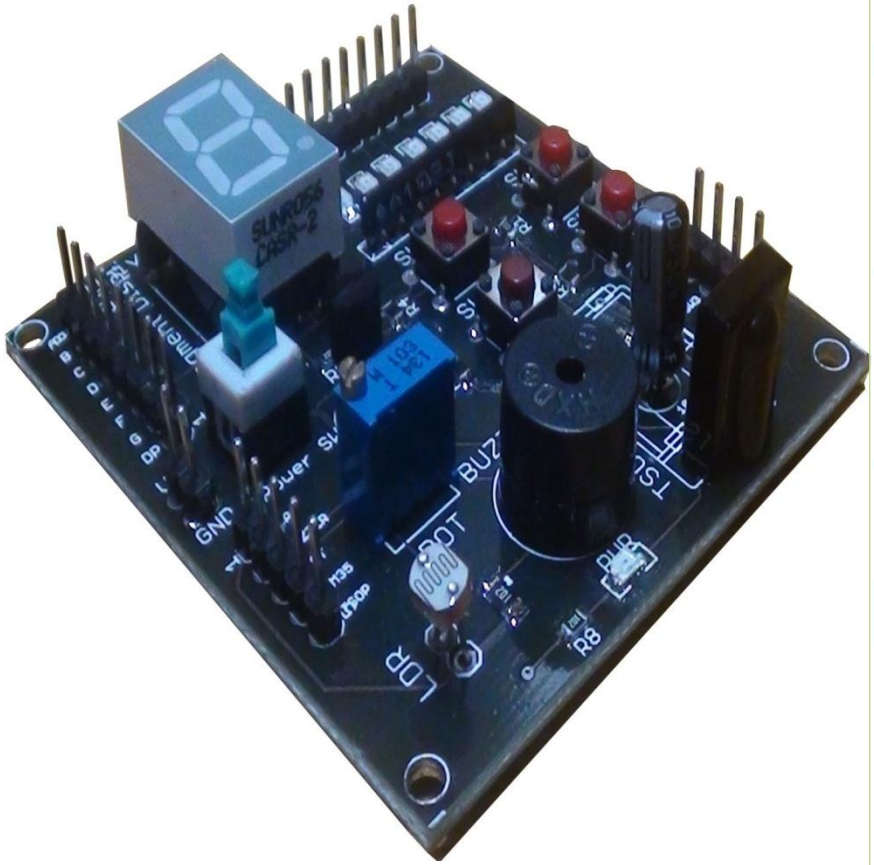
TENET TECHNETRONICS

2015

Universal GPIO Board User Manual V1.0 With Arduino Uno R3



Author: Prajwal

Reviewed by:

Version: 1.0

Revision History

Version	Date	Description	Remarks
V1.0		Initial Version	

Table of contents

1. Universal GPIO Board Introduction

1.1 Overview

Ready to use Input and Output circuits are always important to experiment with any microcontroller. The Universal GPIO Board is very useful for beginners, hobbyist and students. It is suitable for carrying out quick experiments with any microcontroller and lets you access numerous peripheral devices. It provides access to pins through male connectors for wiring to the microcontroller development board.

1.2 Board Details

This board has below listed interface circuits to work with:

1. 8 LEDs
2. 4 Switches
3. 1 Potentiometer
4. 1 Light Sensor (using Light Dependent Resistor)
5. 1 Temperature Sensor
6. 1 Infrared Receiver
7. 1 Buzzer
8. 1 Seven Segment Display

The above mentioned interface circuits fall into analog/digital/Input/Output as depicted below:

1. Digital Inputs to any microcontroller
   - Four Switches
   - Infrared receiver TSOP 1738

2. Analog Inputs to Any Microcontroller
   - Potentiometer
   - Temperature sensor
   - Light Sensor

3. Use as Digital/Analog Outputs for Any Microcontroller
   - LEDs
   - Buzzer
   - Seven Segment Display

2. Board Design

2.1 Hardware Schematic

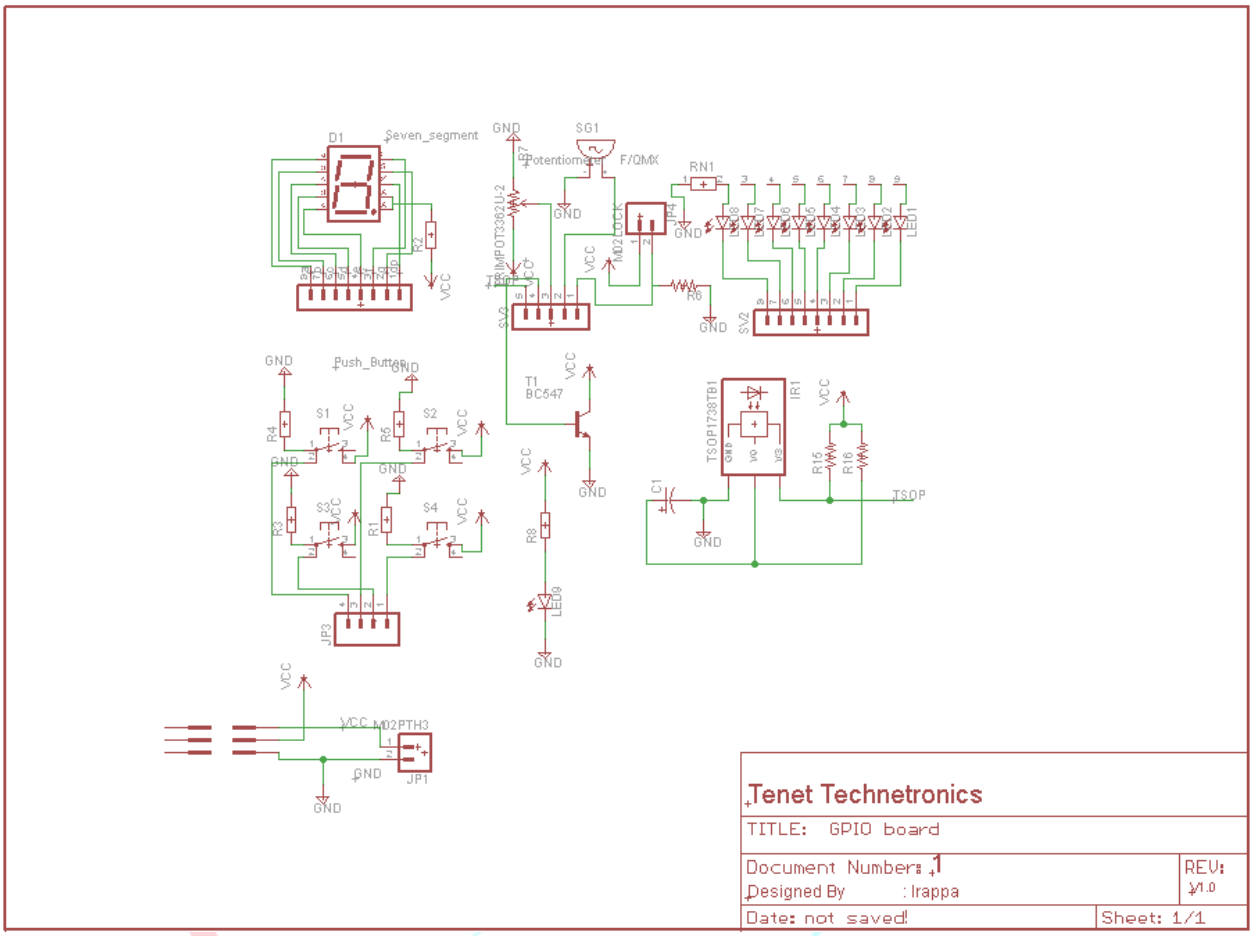


Fig. 1

2.2 Board Layout

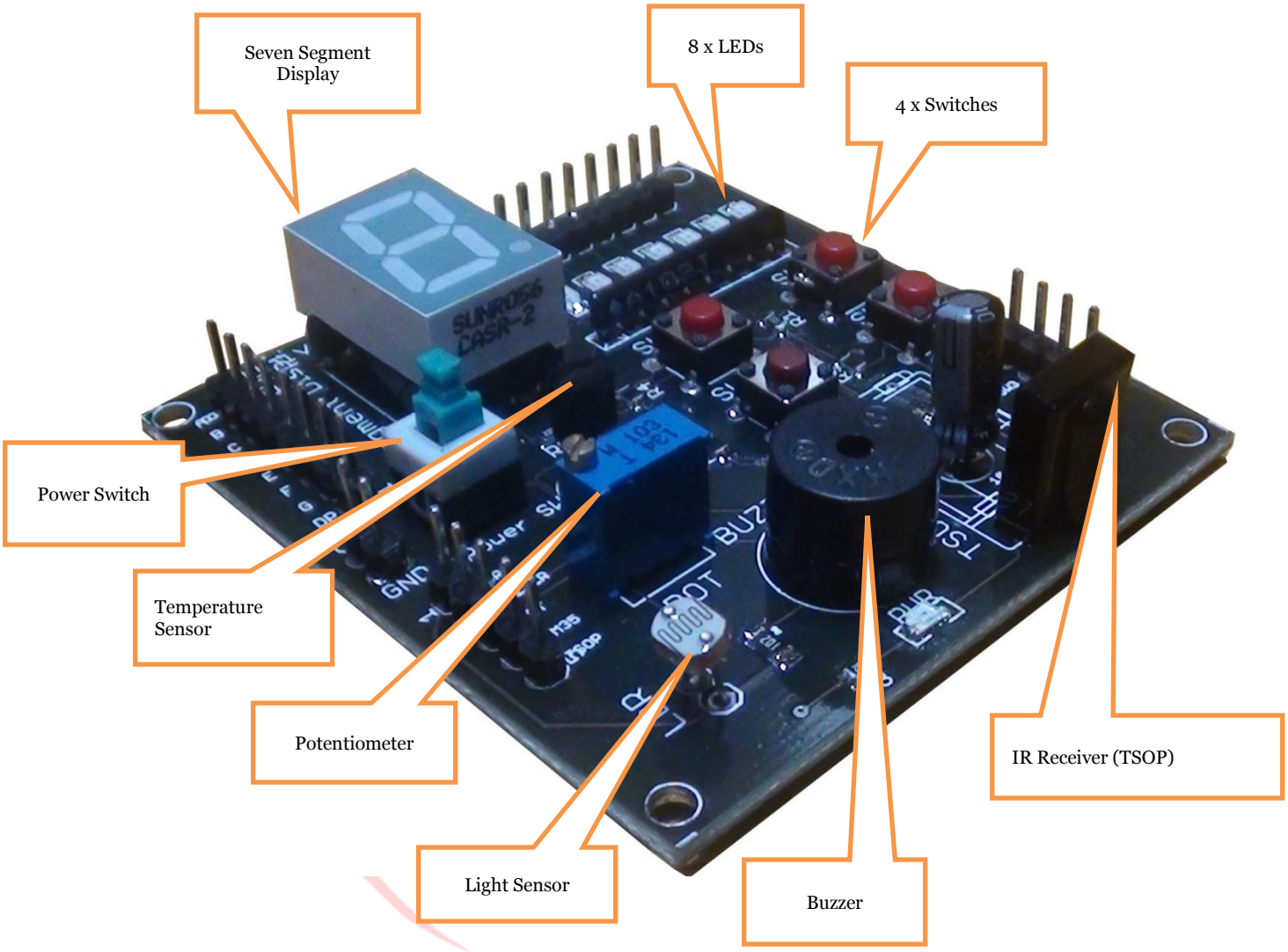


Fig. 2

3. Hardware Details

This section will detail every hardware module, interface parameters and pin definitions of Universal GPIO board.

3.1. Light Emitting Diodes

The Universal GPIO board has eight LEDs, connected to L1, L2…L8 through a network resistor. By driving the pins HIGH (5v), the LEDs can be switched ON.

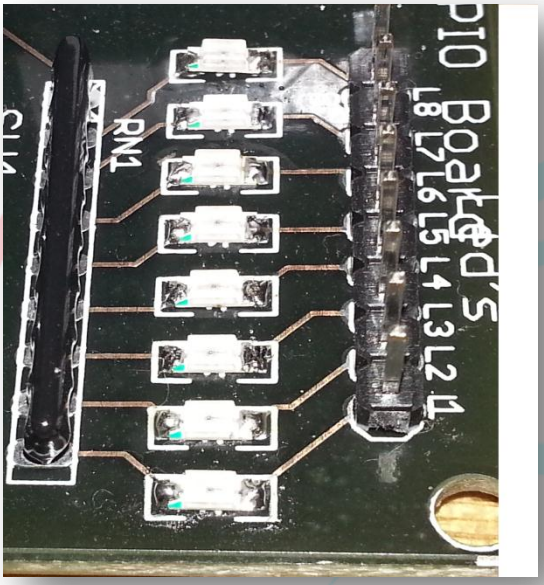


Fig. 3

3.2. Push Buttons

This board provides four mini switch buttons, connected to SW1, SW2, SW3 and SW4. When the pushbutton is open (un pressed) there is no connection between the two legs of the pushbutton, so the pin is connected to GND (through the pull-up resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to ground, so that we read a HIGH.

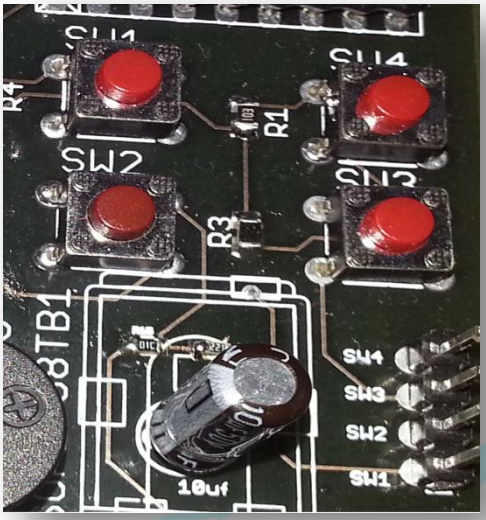


Fig. 4

3.3. Buzzer

This through-hole buzzer is great for projects where you need something that sounds but don't have room for a full-blown speaker. We can access the buzzer from the male connector which is imprinted as Buzzer in the board.

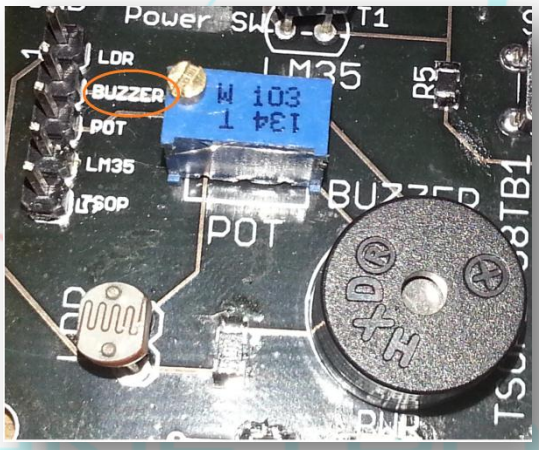


Fig. 5

3.4. Seven Segment Display

The seven segment display is a pretty simple device. It is actually combination of 8 LEDs (the decimal point is the 8th). It can be arranged so that different combinations can be used to make numerical digits.

A seven segment is generally available in ten pin package. While eight pins correspond to the eight LEDs, the remaining two pins (at middle) are common and internally shorted. These segments come in two configurations, namely, Common cathode (CC) and Common anode (CA). In CC

configuration, the negative terminals of all LEDs are connected to the common pins. The common is connected to ground and a particular LED glows when its corresponding pin is given high. In CA arrangement, the common pin is given a high logic and the LED pins are given low to display a number.
This board comes with common anode seven segment displays with the male connector. Use resistor that won't destroy led. We have connected led pin 3 and 8 to 5v through 1k resistor because it is common anode display.
Note: Please don't use Common Cathode Display in this board because common pin is connected to 5V.

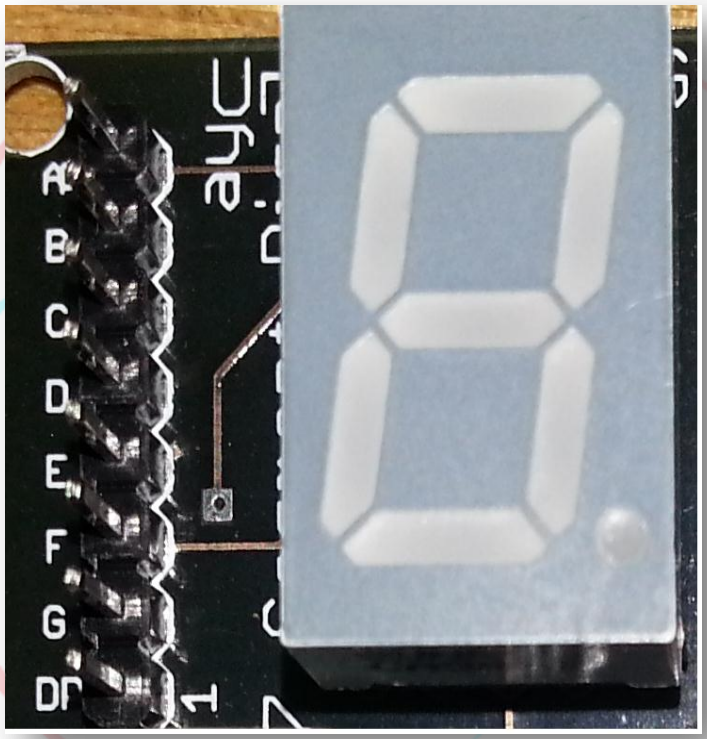


Fig. 6

3.5. Potentiometer

A potentiometer, informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor.
A potentiometer measuring instrument is essentially a voltage divider used for measuring electric potential (voltage); the component is an implementation of the same principle, hence its name.
Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.
This board uses a 10k potentiometer and signal pin is connected to male connector. We can access the signal pin from the male connector which is imprinted as Pot in the board.

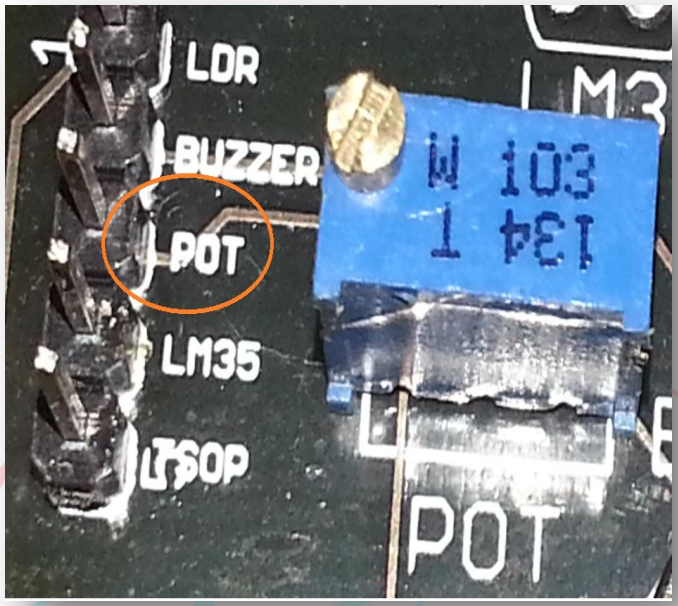


Fig. 7

3.6. Temperature Sensor

The LM35 temperature sensor is a low voltage, precision centigrade temperature sensor. It provides a voltage output that is linearly proportional to the Celsius temperature. It also doesn't require any external calibration. The LM35 is rated to operate over a −55°C to +150°C temperature range, while the LM35C is rated for a −40°C to +110°C range. We like it because it's so easy to use: Just give the device a ground and 2.7 to 5.5 VDC and read the voltage on the LM35 male connector pin. The output voltage can be converted to temperature easily using the scale factor of 10 mV/°C.

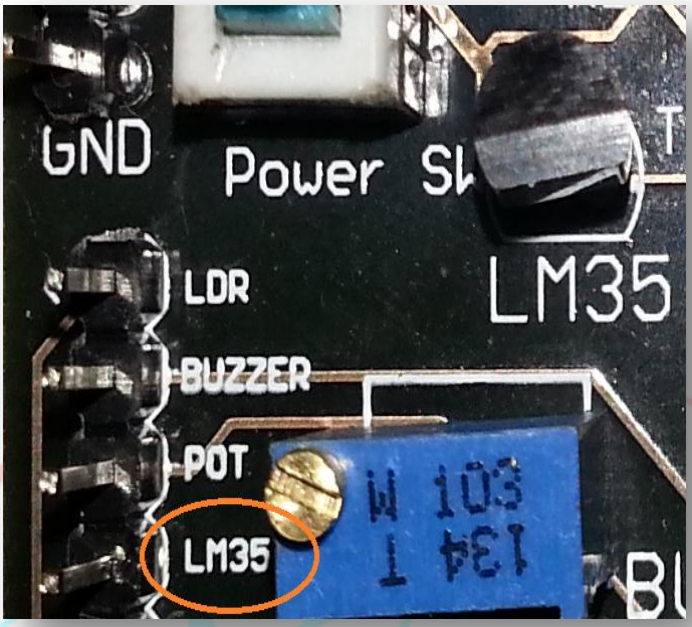


Fig. 8

3.7 IR Receiver (TSOP)

TSOP is an IR receiver which will help you to receive IR signal from transmitting devices like the TV remotes. The TSOP outputs a constant HIGH signal when idle and as it receives data, it tends to invert the data. i.e when an IR LED is transmitting data onto the TSOP, every time the IR led goes high, the TSOP will go LOW and vice versa. Remote control signals are often bytes of data that is encoded and transmitted by pulsing(switching ON & OFF the IR LED at a specific frequency) Most TV remote controls work at 32-40 Khz frequency and most receivers can receive this range.
We can access the TSOP signal pin from the male connector which is imprinted as 'TSOP' in the board.
3.8. Light Sensor
Light-dependent resistor (LDR) or photocell is a light-controlled variable resistor. The resistance of a photo resistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photo resistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits.

Interfacing Universal GPIO Board LED's with Arduino

Let's start with a quick introduction to the Arduino GPIO's (General Purpose Input Output) and number of LED's available in Universal GPIO Board. There are 14 Digital I/O Pins (of which 6 provides PWM output) and 6 Analog Input Pins in Arduino, and There are totally 8 LED's in Universal GPIO Board. Arduino can set to **HIGH** (taking the value 1) by connecting it to a voltage supply, or set to **LOW** (taking the value 0) by connecting it to the ground.

4. LAB Experiments with Arduino

Lab 1:
Blinking All 8 LEDs in GPIO Board

Description:
This experiment shows how to Turns on all LED's ON for one second, then OFF for one second, repeatedly.

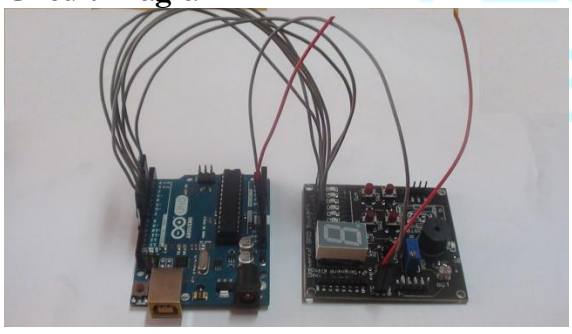
Hardware:
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

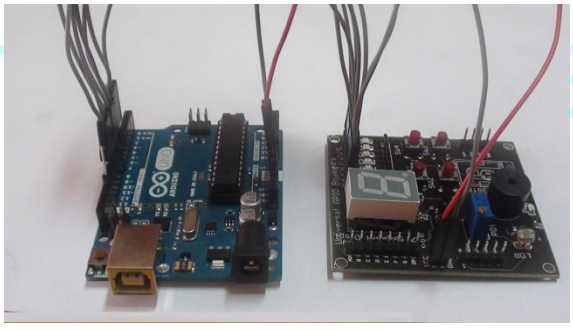
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
5	L8
6	L7
7	L6
8	L5
9	L4
10	L3
11	L2
12	L1

Circuit Diagram





Code:

```
/*
  Blink all LED's
  Turns on all LED's on for one second, then off for one second, repeatedly.
```

```
 */


// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as an output.
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()
{
  // turn on all LED's
  digitalWrite(12, HIGH);     // turn the LED on (HIGH is the voltage level)
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  delay(1000);               // wait for a second

  // turn off all LED's
  digitalWrite(12, LOW);   // turn the LED off by making the voltage LOW
  digitalWrite(11, LOW);
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  digitalWrite(8, LOW);
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);

  delay(1000);               // wait for a second

}
```

Launch Arduino IDE, write the above program, and upload the code to Arduino through USB cable.

Lab 2: Blinking Individual LEDs in GPIO Board

Description:

This experiment shows how to turn on and off one LED at a time in series, repeatedly.

Hardware:

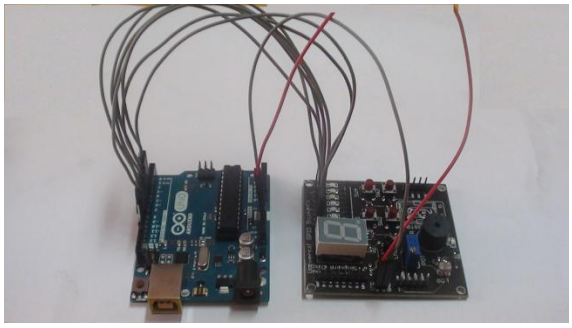
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

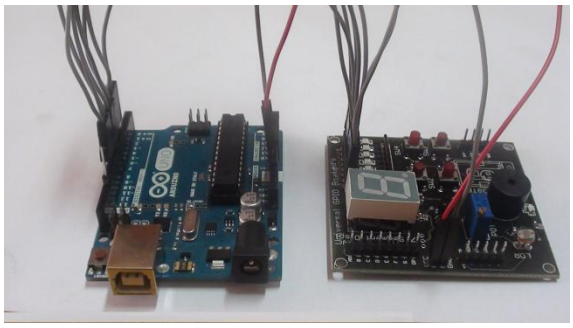
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
5	L8
6	L7
7	L6
8	L5
9	L4
10	L3
11	L2
12	L1

Circuit Diagram





Code:

```
/*
  Blinking individual LED
  Turning ON and OFF one LED at a time in series.
  // connect ARDUINO WITH GPIO BOARD (5,6,7,8,9,10,11,12 to L8,L7,L6,L5,L4,L3,L2,L1)
respectively
    */

// the setup routine runs once when you press reset:
void setup()
{
  // initialize the digital pins as an output.
  for(int i=5; i<=12; i++)
  pinMode(i, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop()
{
  for(int Led=5; Led<=12; Led++)
  {
     digitalWrite(Led, HIGH);   // turn the LED on (HIGH is the voltage level)
     delay(500);
     digitalWrite(Led, LOW);   // turn the LED off by making the voltage LOW
     delay(500);

  }

}
```

Launch Arduino IDE, write the above program, and upload the code to Arduino through USB cable.

Lab 3: Blinking Alternate LEDs in GPIO Board

Description:
This experiment shows how to turn on and off alternating LED, repeatedly.

Hardware:

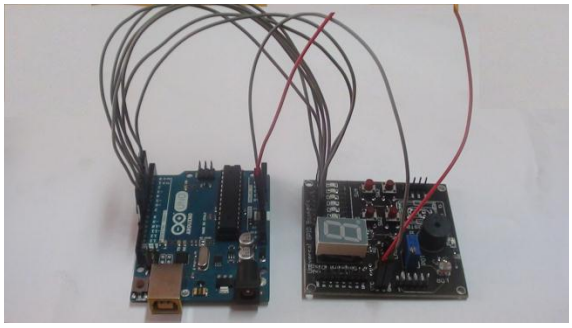
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

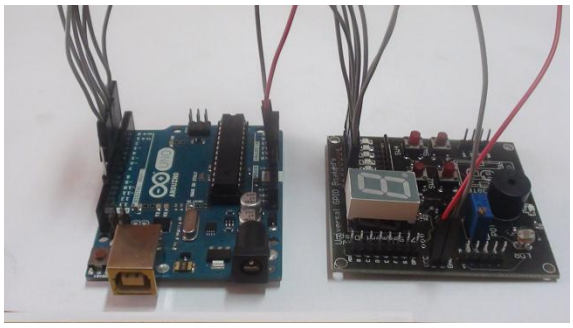
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
5	L8
6	L7
7	L6
8	L5
9	L4
10	L3
11	L2
12	L1

Circuit Diagram





Code:

```
/*
 Alternating Blinking
 Blinking all even and odd LED's alternatingly
 // connect ARDUINO WITH GPIO BOARD (5,6,7,8,9,10,11,12 to L8,L7,L6,L5,L4,L3,L2,L1) respectively
 */

void setup()
{
   // initialize the digital pins as an output.
   for(int i=5; i<=12; i++)
  pinMode(i, OUTPUT);

}
void loop()
{
 // blink all even pins
 digitalWrite(12, HIGH);
 digitalWrite(10, HIGH);
 digitalWrite(8, HIGH);
 digitalWrite(6, HIGH);
 delay(500);              // wait for a second
 digitalWrite(12, LOW);
 digitalWrite(10, LOW);
 digitalWrite(8, LOW);
 digitalWrite(6, LOW);
 delay(500);        // wait for a second

// blink all odd pins
 digitalWrite(11,HIGH);
 digitalWrite(9, HIGH);
 digitalWrite(7, HIGH);
 digitalWrite(5, HIGH);
 delay(500);              // wait for a second
 digitalWrite(11, LOW);
 digitalWrite(9, LOW);
 digitalWrite(7, LOW);
 digitalWrite(5, LOW);
```

```
    delay(500);              // wait for a second

  }
```

Launch Arduino IDE, write the above program, and upload the code to Arduino through USB cable.

Lab 4: Sequencing multiple LEDs in GPIO Board

Description:
This experiment shows how to light up all LED's from Left to Right and turn off the LED's from Right to Left, repeatedly.

Hardware:

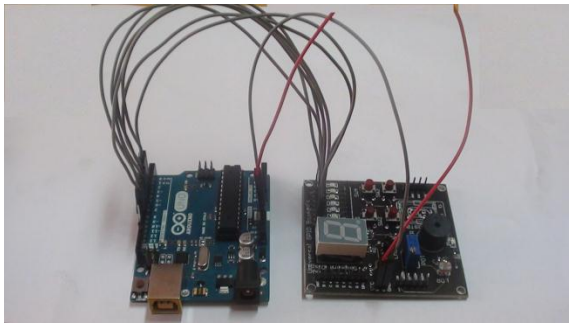
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

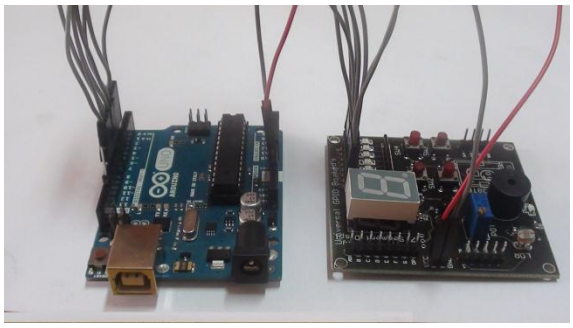
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
5	L8
6	L7
7	L6
8	L5
9	L4
10	L3
11	L2
12	L1

Circuit Diagram





Code:

```
/*
  Blinking in series
 Turning on all LED's Ascending, then turning off all the LED's Descending.
 */

// the setup routine runs once when you press reset:
void setup()
{
  // connect ARDUINO WITH GPIO BOARD (5,6,7,8,9,10,11,12 to L8,L7,L6,L5,L4,L3,L2,L1)
respectively
  // initialize the digital pins as an output.
  for(int i=5; i<=12; i++)
  pinMode(i, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop()
{

  for(int asc=5; asc<=12; asc++)  // Ascending on
  {
  digitalWrite(asc, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);
  }

  delay(1000);     // wait for a second

  for(int des=12; des>=5; des--)  // Descending off
  {
  digitalWrite(des, LOW);   // turn the LED off by making the voltage LOW
  delay(500);
  }

}
```

Launch Arduino IDE, write the above program, and upload the code to Arduino through USB cable.

Lab 5: Fading Alternate LEDs using PWM

Description:
This experiment shows how to fade the contrast of the LED's using PWM.
Note: Since Arduino UNO has only 6 PWM pins, we can only fade 6 LED's

Hardware:

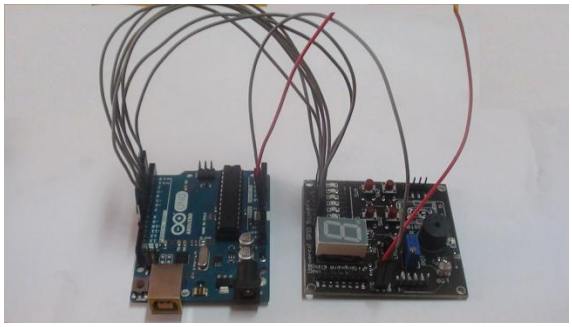
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

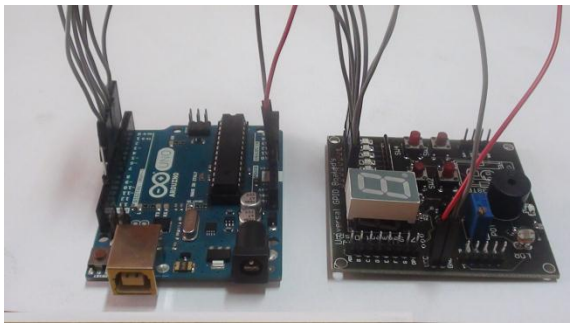
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
3	L8
5	L7
6	L6
9	L5
10	L4
11	L3

Circuit Diagram





Code:

```
/*
 Alternate Fading using pwm
 connect ARDUINO WITH GPIO BOARD (3,5,6,9,10,11 to L8,L7,L6,L5,L4,L3) respectively.
  */

void setup()
{

  // initialize the digital pins as an output.
    for(int i=3; i<=12; i++)
    pinMode(i, OUTPUT);
}
void loop()
{

  for(int fadeValue = 0, fadeValue1 = 255; fadeValue <= 255,fadeValue1>= 0 ; fadeValue
+=5,fadeValue1 -=5)
  {

    analogWrite(3, fadeValue);
    analogWrite(5, fadeValue1);
    analogWrite(6, fadeValue);
    analogWrite(9, fadeValue1);
    analogWrite(10, fadeValue);
    analogWrite(11, fadeValue1);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
   }
   for(int fadeValue = 255, fadeValue1= 0; fadeValue >= 0,fadeValue1 <= 255 ; fadeValue -
=5,fadeValue1 +=5)
  {
 // sets the fadeValue value (range from 255 to 0): and sets the fadeValue1 value (range from 0 to
255)
    analogWrite(3, fadeValue);
    analogWrite(5, fadeValue1);
    analogWrite(6, fadeValue);
    analogWrite(9, fadeValue1);
    analogWrite(10, fadeValue);
```

```
    analogWrite(11, fadeValue1);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
   }
}
```

Launch Arduino IDE, write the above program, and upload the code to Arduino through USB cable.

Lab 6: Incrementing and Decrementing the Count using Pushbuttons in GPIO Board

Description:

This experiment shows how to increment the count when pushbutton1 (SW1) is pressed and decrement the count when pushbutton2 (SW2) is pressed.

Hardware:

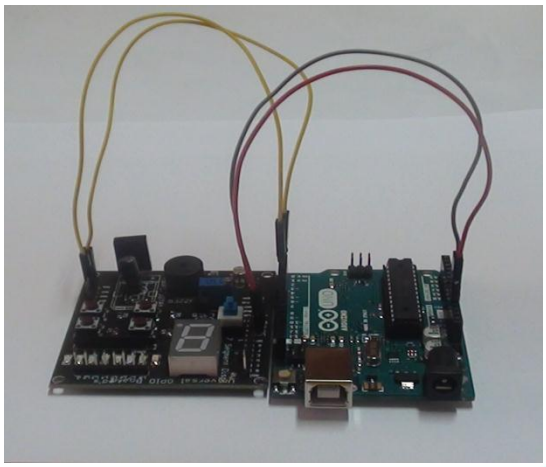
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

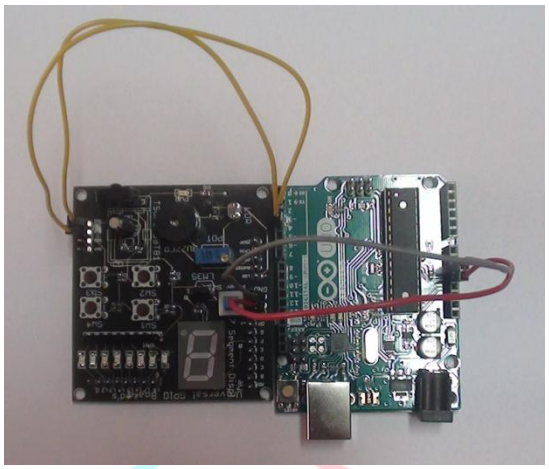
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
2	SW1
3	SW2

Circuit Diagram





Code:

```
/*
  Button
 To Increment the count when pushbutton1 (SW1) is pressed and
 Decrement the count when pushbutton2 (SW2) is pressed.

 */

// set pin numbers:
const int  buttonPin1 = 2;    // the number of the pushbutton1 pin
const int  buttonPin2 = 3;   // the number of the pushbutton12 pin

int buttonState1;             // variable for reading the pushbutton1 status
int buttonState2;            // variable for reading the pushbutton2 status

int buttonPushCounter;      // variable for printing the count status

void setup()
{
  Serial.begin(9600);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);

}


void loop()
{
  // read the state of the pushbutton values:
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);

    // check if the pushbutton1 is pressed.
    // if it is, the buttonState1 is HIGH:
```

```
  if (buttonState1 == HIGH  )
  {
   buttonPushCounter++;
   delay(300);
   Serial.println(buttonPushCounter);
  }
  // check if the pushbutton2 is pressed.
  // if it is, the buttonState2 is HIGH:
  if(buttonState2 == HIGH )
  {
   buttonPushCounter--;
   delay(300);
   Serial.println(buttonPushCounter);
  }

}
```

Launch Arduino IDE, write the above program, upload the code to Arduino through USB cable, open the serial monitor and press the buttons SW1 or SW2 to increment or decrement the count respectively.

Lab 7: Determining which of the 4 Pushbuttons is pressed in the GPIO Board

Description:

This experiment show, which pushbutton is pressed in GPIO Board, which is to determine which of the four pushbutton switches (SW1, SW2, SW3, and SW4), is pressed or high.

Hardware:

- Arduino Board
- USB Cable
- Universal GPIO Board

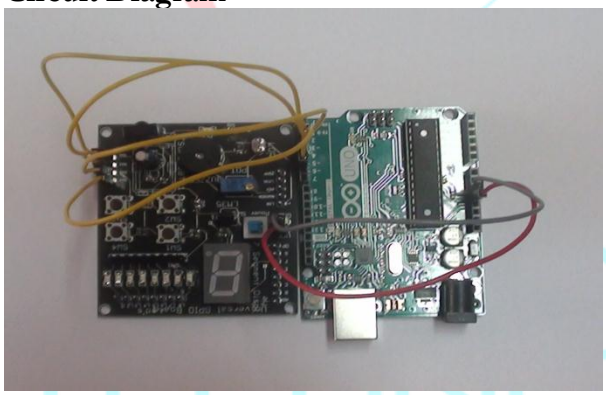
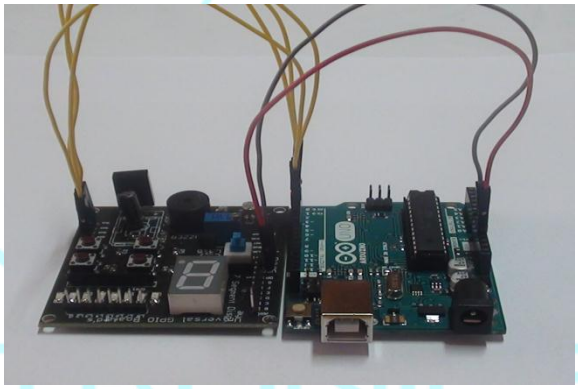
- Jumper wires—Male to Female

If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
2	SW1
3	SW2
4	SW3
5	SW4

Circuit Diagram

Code:

```
/*
  Button
  To know which pushbutton switch (SW1, SW2, SW3, and SW4), is pressed/ high in GPIO Board.
 */


// constants won't change. They're used here to
// set pin numbers:
```

```
const int buttonPin1 = 2;     // the number of the pushbutton1 pin
const int buttonPin2 = 3;     // the number of the pushbutton2 pin
const int buttonPin3 = 4;     // the number of the pushbutton3 pin
const int buttonPin4 = 5;     // the number of the pushbutton4 pin

// variables will change:
int buttonState1 = 0;         // variable for reading the pushbutton1 status
int buttonState2 = 0;         // variable for reading the pushbutton2 status
int buttonState3 = 0;         // variable for reading the pushbutton3 status
int buttonState4 = 0;         // variable for reading the pushbutton4 status

void setup()
{
  Serial.begin(9600);
 // initialize the pushbutton pins as an input:
pinMode(buttonPin1, INPUT);
pinMode(buttonPin2, INPUT);
pinMode(buttonPin3, INPUT);
pinMode(buttonPin4, INPUT);
}

void loop()
{
 // read the state of the pushbutton values:
 buttonState1 = digitalRead(buttonPin1);
 buttonState2 = digitalRead(buttonPin2);
 buttonState3 = digitalRead(buttonPin3);
 buttonState4 = digitalRead(buttonPin4);
  // check if the pushbutton1 is pressed.
 // if it is, the buttonState1 is HIGH:
 if (buttonState1 == HIGH)
 {
  Serial.println("Button 1 pressed");
 }
  // check if the pushbutton2 is pressed.
 // if it is, the buttonState2 is HIGH:
 else if (buttonState2 == HIGH)
 {
  Serial.println("Button 2 pressed");
 }
  // check if the pushbutton3 is pressed.
 // if it is, the buttonState3 is HIGH:
 else if (buttonState3 == HIGH)
 {
  Serial.println("Button 3 pressed");
 }
  // check if the pushbutton4 is pressed.
 // if it is, the buttonState4 is HIGH:
```

```
  else if (buttonState4 == HIGH)
  {
    Serial.println("Button 4 pressed");
  }
}
```

Launch Arduino IDE, write the above program, upload the code to Arduino through USB cable, open the serial monitor and press the buttons SW1, SW2, SW3, and SW4 to identify which of the button is pressed.

Lab 8: Making Beep Sound

Description:

There is a Buzzer in the GPIO board and we will generate or initiate a constant tone or a Beep sound in the Buzzer. This experiment shows how to produce a Beep sound for one second and make OFF for one second.

Hardware:

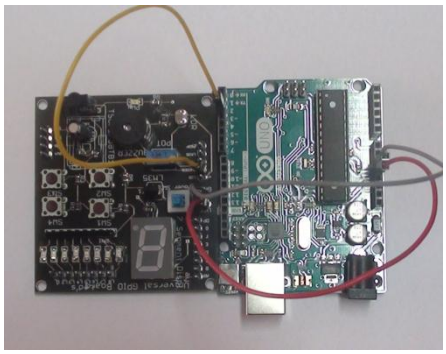
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

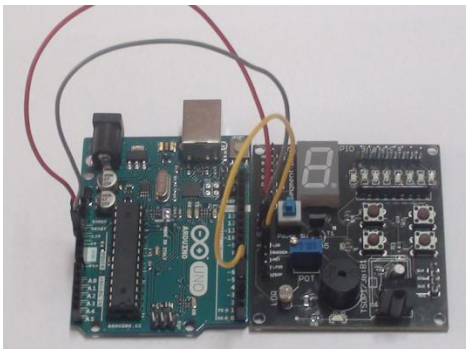
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
2	BUZZER

Circuit Diagram





Code:

```
/*
  Making Beep Sound
  Turn on a Buzzer for one second, then off for one second, repeatedly.

 */

// the setup routine runs once when you press reset:
void setup()
{
  // initialize the digital pin as an output.
  pinMode(2, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()
{
  digitalWrite(2, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(2, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Launch Arduino IDE, write the above program, and upload the code to Arduino through USB cable.

Lab 9: Playing a Simple Melody

Description:

This experiment shows how to play a simple melody with an Arduino and a buzzer. Tone is used to play a melody on a buzzer. It generates a melody on the specified pin, with the required frequency and duration. If you do not specify the duration, it keeps playing until the no Tone tag. Using Tone will affect PWM outputs of digital pins 3 and 11. Only one tone can be generated at a time.

Hardware:

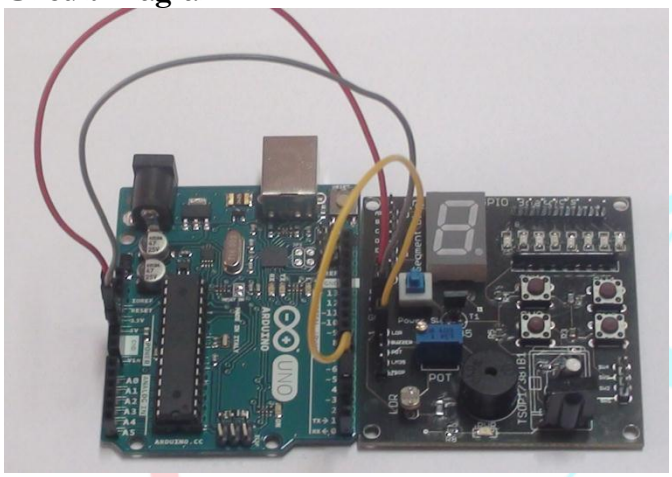
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
8	BUZZER

Circuit Diagram



Code:

```
/*
  Melody

 Plays a melody

 circuit:
 * Buzzer on digital pin 8

This example code is in the public domain.
  http://arduino.cc/en/Tutorial/Tone

 */
 #include "pitches.h"

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4,4,4,4,4 };

void setup() {
```

```
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];
    tone(8, melody[thisNote],noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

Note: Copy the below file and save it as pitches.h to the same folder as the Arduino sketch. Failing to do so will result in compile errors.

```
/*************************************************
 * Public Constants
 * See http://en.wikipedia.org/wiki/Music_notes
 * for info on note frequencies
 * Based on http://arduino.cc/en/Tutorial/Tone
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
```

```
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
```

```
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

Launch Arduino IDE, write the above program, and upload the code to Arduino through USB cable. Make sure that you place a copy of the "pitches.h" file in the same directory, or you will get a compile error from the Arduino IDE. The "pitches.h" file contains a mapping of the most common musical notes to their respective frequency.

Lab 10: Detecting Pot Values on Serial Monitor

Description:

This experiment shows how to read the analog values using Potentiometer and display in serial monitor.

Hardware:

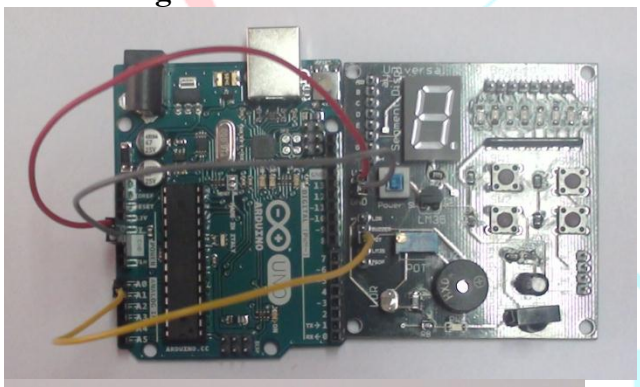
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

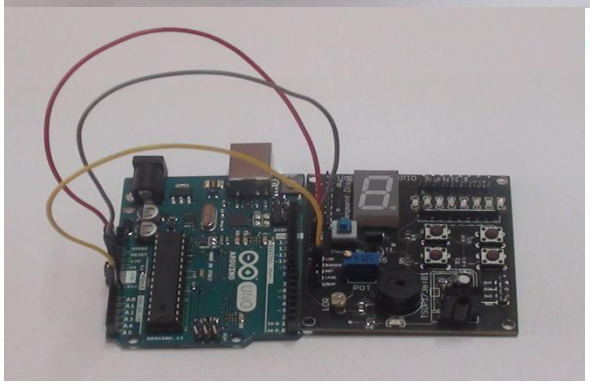
If you do not have any of these components, make online order from the following link. www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
A0	POT

Circuit Diagram





Code:

```
/*
  Detecting Pot Values on Serial Monitor

  */
int sensorPin = A0;    // select the input pin for the potentiometer
int PotValue = 0;  // variable to store the value coming from the POT

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

void loop() {
 // read the value from the sensor:
  PotValue = analogRead(sensorPin);
 // print the POT values in serial monitor:
  Serial.println(PotValue);
}
```

Lab 11: Plotting the Pot Values on Processing

Description:

This experiment shows how to read the analog values using Potentiometer and display in Processing.

Hardware:

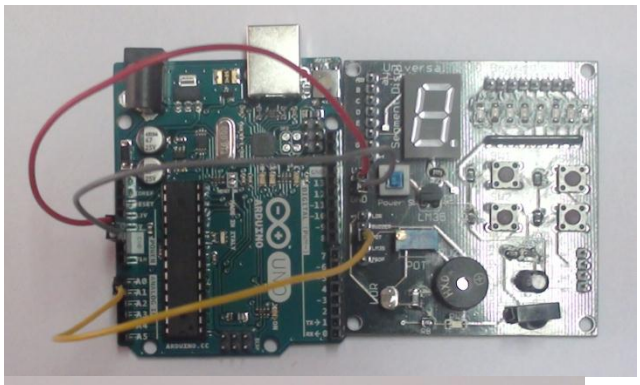
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

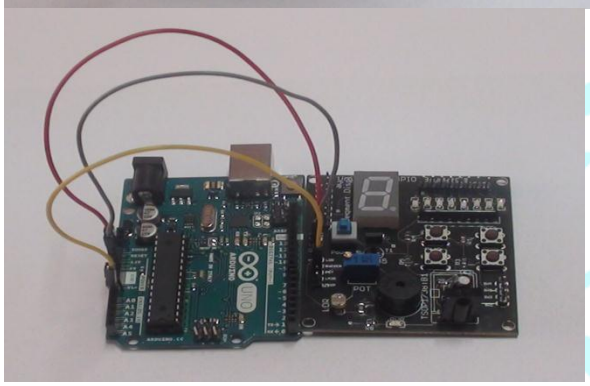
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
A0	POT

Circuit Diagram





Code:

Load this code onto the Arduino:

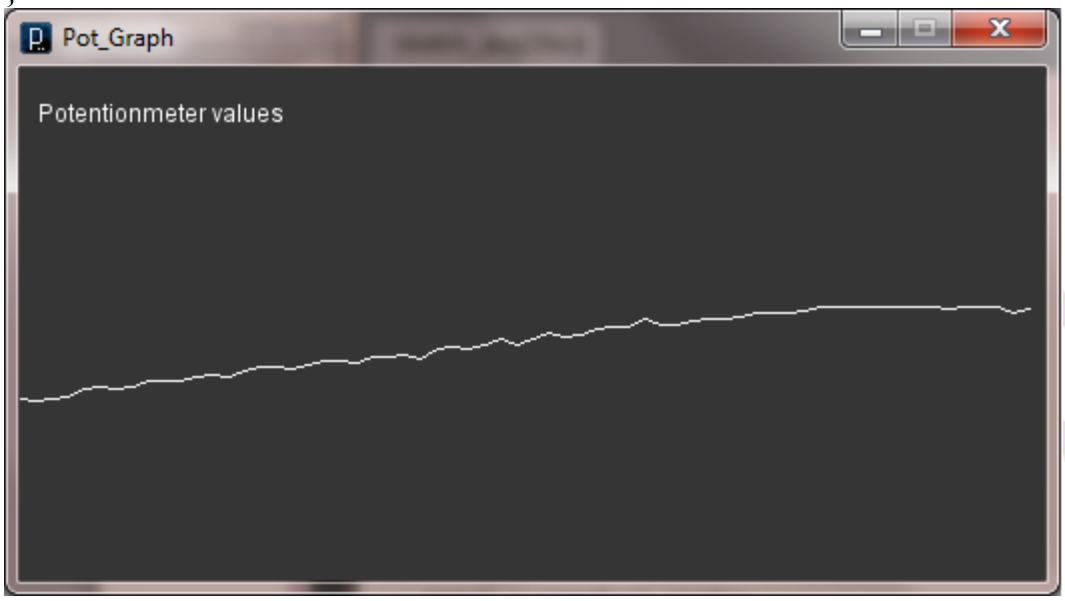
```
/*
  Detecting Pot Values on Serial Monitor


  */
int sensorPin = A0;    // select the input pin for the potentiometer
int PotValue = 0;  // variable to store the value coming from the POT

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

}

void loop() {
 // read the value from the sensor:
  PotValue = analogRead(sensorPin);
 // print the POT values in serial monitor:
  Serial.println(PotValue);
  delay(100);
}
```

Then use this code in Processing:

```
// Demonstrates reading data from the Arduino board by graphing the
// values received.
//
```

```
// based on Analog In

import processing.serial.*;
Serial port;
String buff = "";
int NEWLINE = 10;
// Store the last 64 values received so we can graph them.
int[] values = new int[64];
PFont font24;
void setup()
{
  //setup fonts for use throughout the application
  font24 = loadFont("Verdana-24.vlw");
   size(512, 256);
// Change appropriate serial port no in the below line
 port = new Serial(this, "COM26", 9600);

}
void draw()
{
 background(53);
 stroke(255);
 // Graph the stored values by drawing a lines between them.
 for (int i = 0; i < 63; i++)
 line(i * 8, 255 - values[i], (i + 1) * 8, 255 - values[i + 1]);
 while (port.available() > 0)
 {
   text("Potentionmeter values", 5, 25);
   serialEvent(port.read());
 }
}
void serialEvent(int serial)
{
 if (serial != NEWLINE) {
 // Store all the characters on the line.
 buff += char(serial);
 } else {
 // The end of each line is marked by two characters, a carriage
 // return and a newline. We're here because we've gotten a newline,
 // but we still need to strip off the carriage return.
 buff = buff.substring(0, buff.length()-1);
 // Parse the String into an integer. We divide by 4 because
 // analog inputs go from 0 to 1023 while colors in Processing
 // only go from 0 to 255.
 int val = Integer.parseInt(buff)/4;
 // Clear the value of "buff"
 buff = "";
 // Shift over the existing values to make room for the new one.
```

```
for (int i = 0; i < 63; i++)
values[i] = values[i + 1];
// Add the received value to the array.
values[63] = val;
}
}
```



You should get a quickly scrolling graph that looks something like this in the above graph:

Lab 12: Sensing Light using LDR

Description:

This experiment shows of sensing light using LDR sensor and display it in Serial monitor.

Hardware:

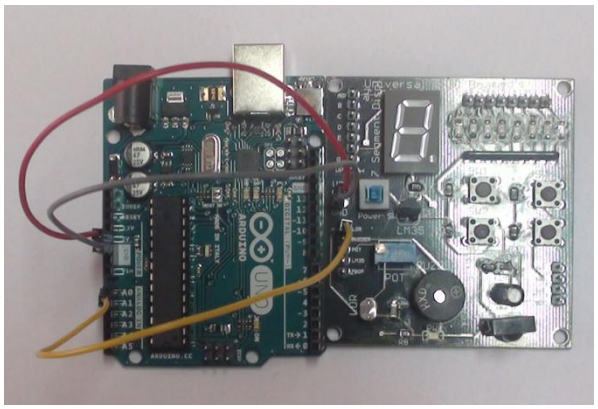
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

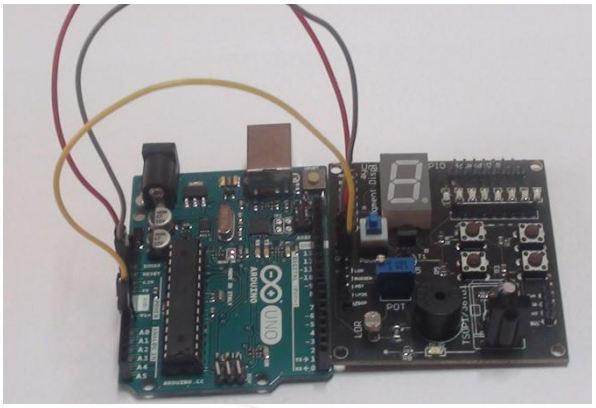
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
A0	LDR

Circuit Diagram





Code:

```
/*
  Sensing Light Values on Serial Monitor


  */
int sensorPin = A0;    // select the input pin for the potentiometer
int sensorValue = 0;  // variable to store the value coming from the sensor

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

}

void loop() {
 // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
 // print the sensor values in serial monitor:
  Serial.println(sensorValue);
  delay(100);
}
```

Launch Arduino IDE, write the above program, upload the code to Arduino through USB cable, and open the serial monitor

Lab 13: Plotting the LDR Values on Processing

Description:

This experiment shows of sensing light using LDR sensor and display it in Serial monitor.

Hardware:

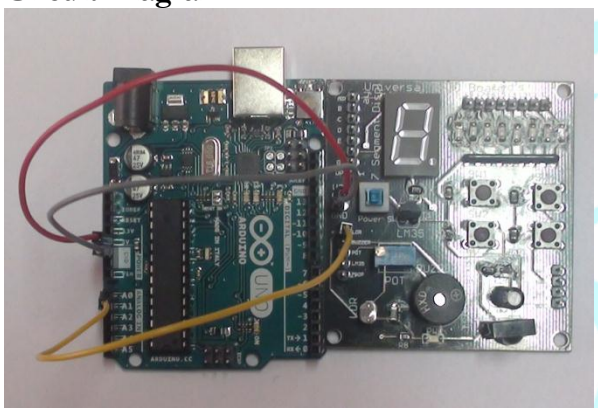
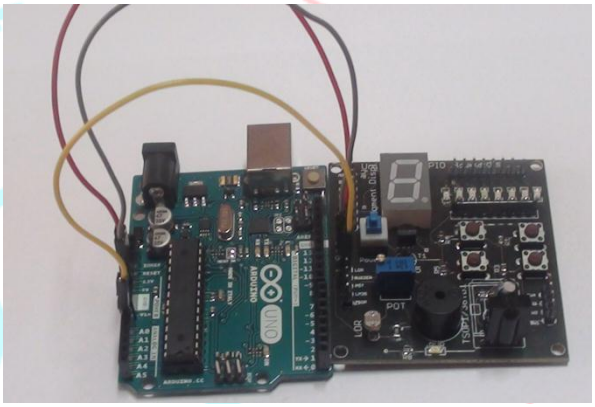
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
A0	LDR

Circuit Diagram

Code:

Load this code onto the Arduino:

```
/*
  Sensing Light Values on Serial Monitor


  */
int sensorPin = A0;    // select the input pin for the potentiometer
int sensorValue = 0;  // variable to store the value coming from the sensor

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

}

void loop() {
 // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
 // print the sensor values in serial monitor:
  Serial.println(sensorValue);
  delay(100);
}
```

Then use this code in Processing:

```
// Demonstrates reading data from the Arduino board by graphing the
// values received.
```

```
//
// based on Analog In

import processing.serial.*;
Serial port;
String buff = "";
int NEWLINE = 10;
// Store the last 64 values received so we can graph them.
int[] values = new int[64];
void setup()
{
 size(512, 256);
 println("Available serial ports:");
 println(Serial.list()); // Uses the first port in this list (number 0). Change this to
 // select the port corresponding to your Arduino board. The last
 // parameter (e.g. 9600) is the speed of the communication. It
 // has to correspond to the value passed to Serial.begin() in your
 // Arduino sketch.
 port = new Serial(this, Serial.list()[0], 9600);
 // If you know the name of the port used by the Arduino board, you
 // can specify it directly like this.
 //port = new Serial(this, "COM1", 9600);
}
void draw()
{
 background(53);
 stroke(255);
 // Graph the stored values by drawing a lines between them.
 for (int i = 0; i < 63; i++)
 line(i * 8, 255 - values[i], (i + 1) * 8, 255 - values[i + 1]);
 while (port.available() > 0)
 serialEvent(port.read());
}
void serialEvent(int serial)
{
 if (serial != NEWLINE) {
 // Store all the characters on the line.
 buff += char(serial);
 } else {
 // The end of each line is marked by two characters, a carriage
 // return and a newline. We're here because we've gotten a newline,
 // but we still need to strip off the carriage return.
 buff = buff.substring(0, buff.length()-1);
 // Parse the String into an integer. We divide by 4 because
 // analog inputs go from 0 to 1023 while colors in Processing
 // only go from 0 to 255.
 int val = Integer.parseInt(buff)/4;
 // Clear the value of "buff"
```

```
buff = "";
// Shift over the existing values to make room for the new one.
for (int i = 0; i < 63; i++)
values[i] = values[i + 1];
// Add the received value to the array.
values[63] = val;
}
}
```

Lab 14: Display the Temperature Values on Serial Monitor

Description:

This example shows how to monitor the temperature of a surrounding area and plotting it on processing. Here we can use "PROCESSING Software for plotting the temperature values.

Hardware:

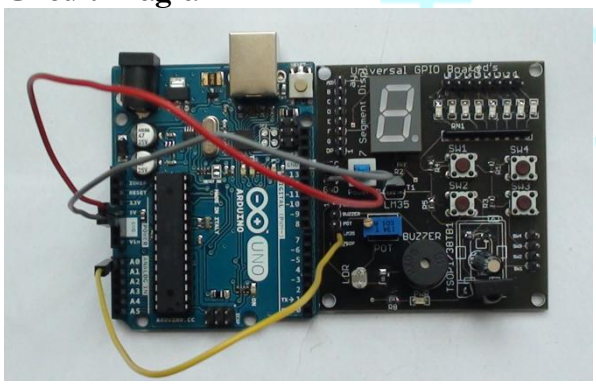
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

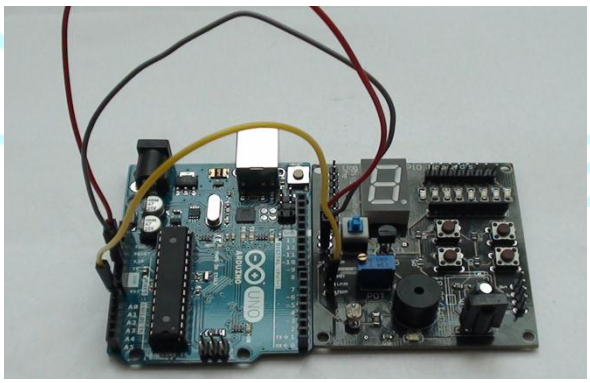
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
A0	LM35

Circuit Diagram





Arduino Code:

```
//declare variables
float tempC;
int tempPin = 0;
```

```
void setup()
{
 Serial.begin(9600); //opens serial port, sets data rate to 9600 bps
}

void loop()
{
 tempC = analogRead(tempPin);          //read the value from the sensor
 tempC = (5.0 * tempC * 100.0)/1024.0;  //convert the analog data to temperature
 Serial.print((byte)tempC);            //send the data to the computer
 delay(1000);                          //wait one second before sending new data
}
```

Lab 15: Plotting the Temperature Values on Processing

Description:
This example shows how to monitor the temperature of a surrounding area and plotting it on processing. Here we can use "PROCESSING Software for plotting the temperature values.

Hardware:

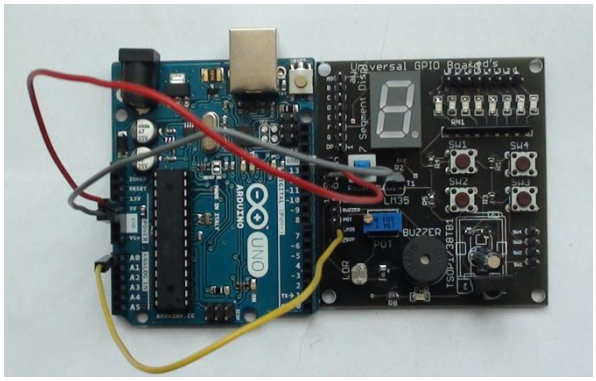
- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

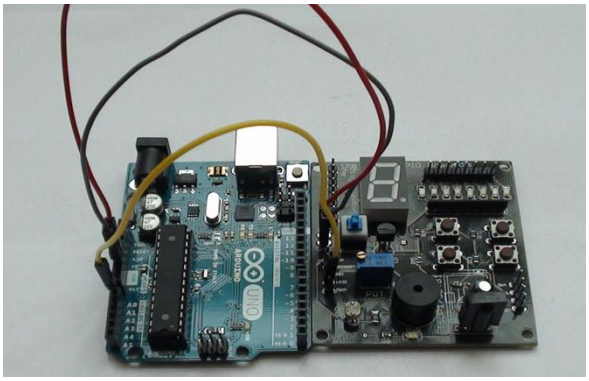
If you do not have any of these components, make online order from the following link.
www.tenettech.com

Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
A0	LM35

Circuit Diagram





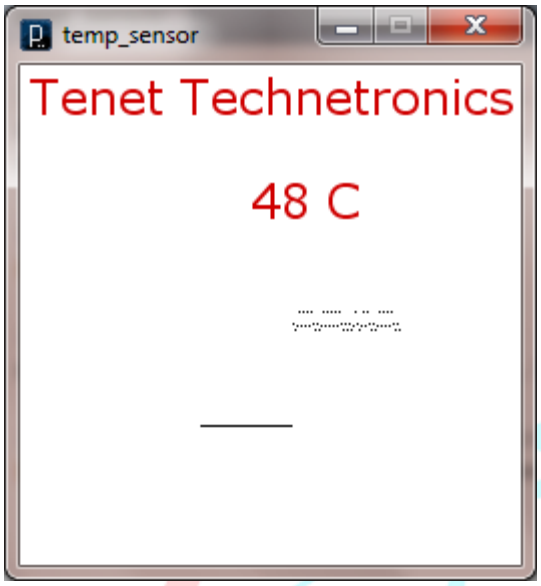
Arduino Code:

```
//declare variables
float tempC;
int tempPin = 0;

void setup()
{
  Serial.begin(9600); //opens serial port, sets data rate to 9600 bps
}

void loop()
{
  tempC = analogRead(tempPin);          //read the value from the sensor
  tempC = (5.0 * tempC * 100.0)/1024.0;  //convert the analog data to temperature
  Serial.print((byte)tempC);            //send the data to the computer
  delay(1000);                          //wait one second before sending new data
}
```

Processing Code:

```
//import Serial communication library
import processing.serial.*;

//init variables
Serial commPort;
float tempC;
float tempF;
int yDist;
PFont font12;
PFont font24;
float[] tempHistory = new float[100];

void setup()
{
  //setup fonts for use throughout the application
  font12 = loadFont("Verdana-12.vlw");
  font24 = loadFont("Verdana-24.vlw");

  //set the size of the window
  size(250, 250);
```

```
  //init serial communication port
  commPort = new Serial(this, "COM6", 9600);

  //fill tempHistory with default temps
  for(int index = 0; index<100; index++)
    tempHistory[index] = 0;
}

void draw()
{
  //get the temp from the serial port
  while (commPort.available() > 0)
  {
    tempC = commPort.read();

    //refresh the background to clear old data
    background(160);
    text("Tenet Technetronics", 5, 25);
    //draw the temp rectangle
    colorMode(RGB, 160);  //use color mode sized for fading



    //draw graph
    stroke(0);
    fill(255,255,255);
    //rect(90,80,100,100);
    for (int index = 0; index<100; index++)
    {
      if(index == 99)
        tempHistory[index] = tempC;
      else
        tempHistory[index] = tempHistory[index + 1];
      point(90 + index, 180 - tempHistory[index]);
    }




    //write the temp in C and F
    fill(130,0,0);
    textFont(font24);
    textAlign(LEFT);
    text(str(int(tempC)) + " C", 115, 77);

  }
}
```



Lab 16: Counting Digits from 0-9

Description:
This example shows how to drive a common anode seven segment display with Arduino.

Hardware:

- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

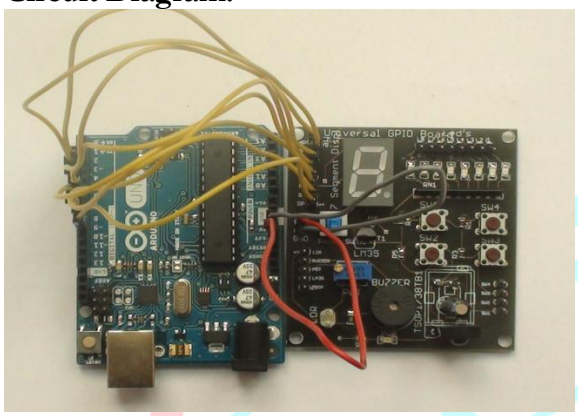
If you do not have any of these components, make online order from the following link.
www.tenettech.com

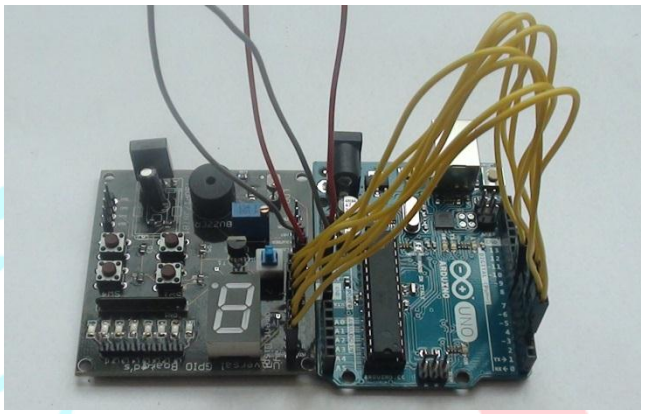
Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
2	A
3	B
4	C
5	D
6	E
7	F
8	G

9	**DP**

Circuit Diagram:





Code:

```
// Define the LED digit patters, from 0 - 9
// Note that these patterns are for common anode displays
// For common cathode displays, change the 0's to 1's and 1's to 0's
// 0 = LED on, 1 = LED off, in this order:
//                          Arduino pin: 2,3,4,5,6,7,8
byte digits[10][7] = { { 0,0,0,0,0,0,1 },  // = 0
                       { 1,0,0,1,1,1,1 },  // = 1
                       { 0,0,1,0,0,1,0 },  // = 2
                       { 0,0,0,0,1,1,0 },  // = 3
                       { 1,0,0,1,1,0,0 },  // = 4
                       { 0,1,0,0,1,0,0 },  // = 5
                       { 0,1,0,0,0,0,0 },  // = 6
                       { 0,0,0,1,1,1,1 },  // = 7
                       { 0,0,0,0,0,0,0 },  // = 8
                       { 0,0,0,1,1,0,0 }   // = 9
                                 };

void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
```

```
  pinMode(9, OUTPUT);
  writeDot(0);  // start with the "dot" off
}

void writeDot(byte dot)
{
  digitalWrite(9, dot);
}
void loop() {

  for (byte count = 0; count < 10; ++count)
  {
    delay(1000);
    byte pin = 2;
    for (byte segCount = 0; segCount < 7; ++segCount)
    {
      digitalWrite(pin, digits[count][segCount]);
      ++pin;
    }
  }
 delay(4000);
}
```

Lab 17: Controlling digits from Serial Monitor:

Description:

This example shows how to drive a common anode seven segment display with Arduino.

Hardware:

- Arduino Board
- USB Cable
- Universal GPIO Board
- Jumper wires—Male to Female

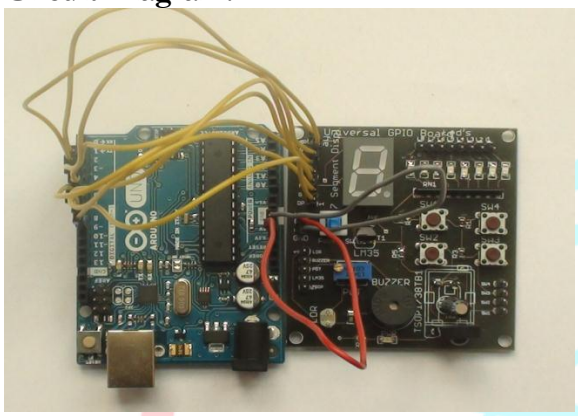
If you do not have any of these components, make online order from the following link.
www.tenettech.com

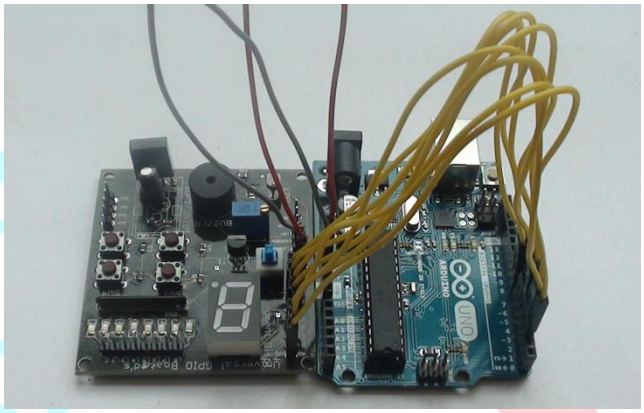
Connection:

Arduino Board pins	Universal GPIO Board pins
5V	VCC
GND	GND
2	A
3	B
4	C
5	D
6	E

7	**F**
8	**G**
9	**DP**

Circuit Diagram:





Code:

```
// Define the LED digit patters, from 0 - 9
// Note that these patterns are for common anode displays
// For common cathode displays, change the 0's to 1's and 1's to 0's
// 0 = LED on, 1 = LED off, in this order:
//                              Arduino pin: 2,3,4,5,6,7,8
int digits[10][7] = { { 0,0,0,0,0,0,1 },  // = 0
                      { 1,0,0,1,1,1,1 },  // = 1
                      { 0,0,1,0,0,1,0 },  // = 2
                      { 0,0,0,0,1,1,0 },  // = 3
                      { 1,0,0,1,1,0,0 },  // = 4
                      { 0,1,0,0,1,0,0 },  // = 5
                      { 0,1,0,0,0,0,0 },  // = 6
                      { 0,0,0,1,1,1,1 },  // = 7
                      { 0,0,0,0,0,0,0 },  // = 8
                      { 0,0,0,1,1,0,0 }   // = 9
                                  };

void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  writeDot(0);  // start with the "dot" off
```

```
}

void writeDot(int dot)
{
 digitalWrite(9, dot);
}
void loop()
{
 char digit;
 int pin;
 if(Serial.available())
 {
  digit = Serial.read();
 }
 switch(digit)
 {
  case '0':
  pin = 2;
  for (int segCount = 0; segCount < 7; ++segCount)
  {
   digitalWrite(pin, digits[digit][segCount]);
   ++pin;
  }
  break;

  case '1':
  pin = 2;
  for (int segCount = 0; segCount < 7; ++segCount)
  {
   digitalWrite(pin, digits[digit][segCount]);
   ++pin;
  }
  break;

  case '2':
  //int pin = 2;
  for (int segCount = 0; segCount < 7; ++segCount)
  {
   digitalWrite(pin, digits[digit][segCount]);
   ++pin;
  }
  break;

  case '3':
  pin = 2;
  for (int segCount = 0; segCount < 7; ++segCount)
  {
   digitalWrite(pin, digits[digit][segCount]);
```

```
  ++pin;
}
break;

case '4':
pin = 2;
for (int segCount = 0; segCount < 7; ++segCount)
{
  digitalWrite(pin, digits[digit][segCount]);
  ++pin;
}
break;

case '5':
pin = 2;
for (int segCount = 0; segCount < 7; ++segCount)
{
  digitalWrite(pin, digits[digit][segCount]);
  ++pin;
}
break;

case '6':
pin = 2;
for (int segCount = 0; segCount < 7; ++segCount)
{
  digitalWrite(pin, digits[digit][segCount]);
  ++pin;
}
break;
case '7':
pin = 2;
for (int segCount = 0; segCount < 7; ++segCount)
{
  digitalWrite(pin, digits[digit][segCount]);
  ++pin;
}
break;

case '8':
pin = 2;
for (int segCount = 0; segCount < 7; ++segCount)
{
  digitalWrite(pin, digits[digit][segCount]);
  ++pin;
}
break;
```

```
    case '9':
    pin = 2;
    for (int segCount = 0; segCount < 7; ++segCount)
    {
      digitalWrite(pin, digits[digit][segCount]);
      ++pin;
    }
    break;
  }
}
```

For more information please visit: www.tenettech.com

For technical query please send an e-mail: info@tenettech.com