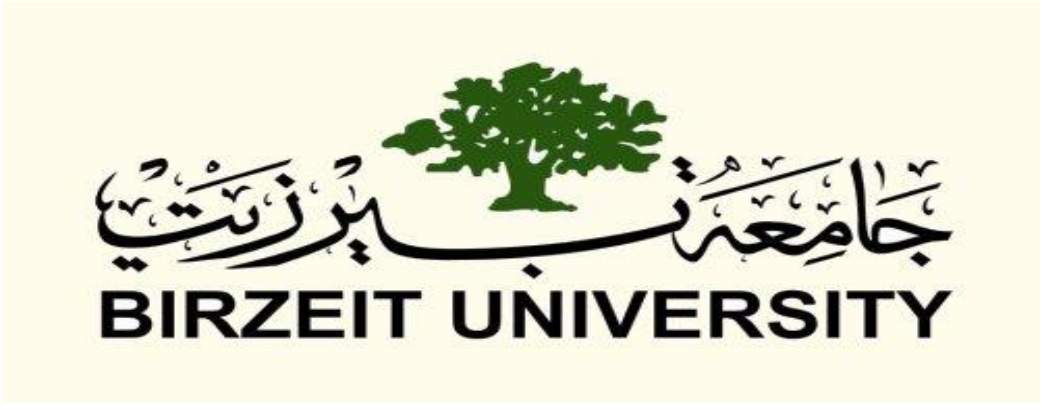
بسم الله الرحمن الرحيم



Faculty of Engineering & Information Technology

Interfacing Laboratory - ENCS 412

Experiments 4-6 Report

LabVIEW, Processing & MBED

Instructor: Dr. Ahmad Afaneh

Teacher Assistant: Eng. Mohammed Modallal

Name: Hussein Dahir, **ID:** 1131138

Partner's Name: Yazeed Obaid, **ID:** 1130036

Partner's Name: Maher Saleem, **ID:** 1130258

Partner's Name: Majd Bassoumi, **ID:** 1130018

Section: 2

Date: 11 April 2017

Abstract:

In these three experiments, More LabVIEW applications was introduced, in addition to introducing the processing program, which is an environment for people who want to program images, animation, and interactions, and it is an open source programming language. Moreover, the MBED microcontroller was introduced, alongside with many examples that help the student to get familiar with it.

Contents

Theory:

LabVIEW [1]:

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a graphical programming environment. So it relies on graphical components to implement different systems and different programing scripts. There are many applications use the Lab-View in different fields: science, education and industry. In this Experiment, the students will write programs to become familiar with some of the basics of LabVIEW.

Processing [2]:

What is Processing?

PROCESSING is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Processing has come to be used for more advanced production-level work in addition to its sketching role. Originally built as a domain-specific extension to Java targeted towards artists and designers, Processing has evolved into a full-blown design and prototyping tool used for large-scale installation work, motion graphics, and complex data visualization. Processing is based on Java, but because program elements in Processing are fairly simple, you can learn to use it even if you don't know any Java.

Why Processing?

There are many reasons Processing makes you more productive when using Arduino:

- Free to download and open source.
- Interactive programs with 2D, 3D, or PDF output.
- OpenGL integration for accelerated 2D and 3D.
- For GNU/Linux, Mac OS X, and Windows
- Over 100 libraries extend the core software.
- Well documented, with many books available.

MBED [3]:

What is MBED?

The MBED platform provides free software libraries, hardware designs and online tools for professional rapid prototyping of products based on ARM microcontrollers. The platform includes a standards-based C/C++ SDK, a microcontroller HDK and supported development boards, an online compiler and online developer collaboration tools.

Why MBED?

1. Easy to transfer the object code: USB link, no need of a specific programmer.
2. Easy to wire on a protoboard: 40 pins DIP package
3. Easy to write a program:
    - Efficient C++ libraries.
    - No need to install any software: online development tool.

- There is offline development tool.
- Community forum and wiki.

MBED module and schematics:

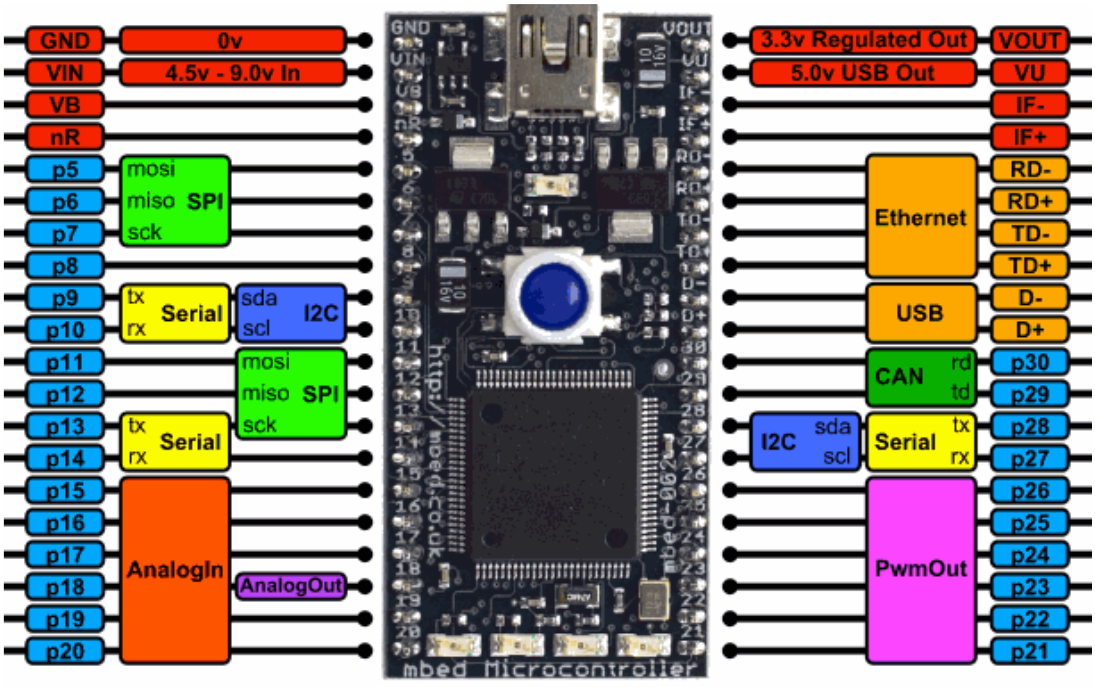


Figure 1: MBED module

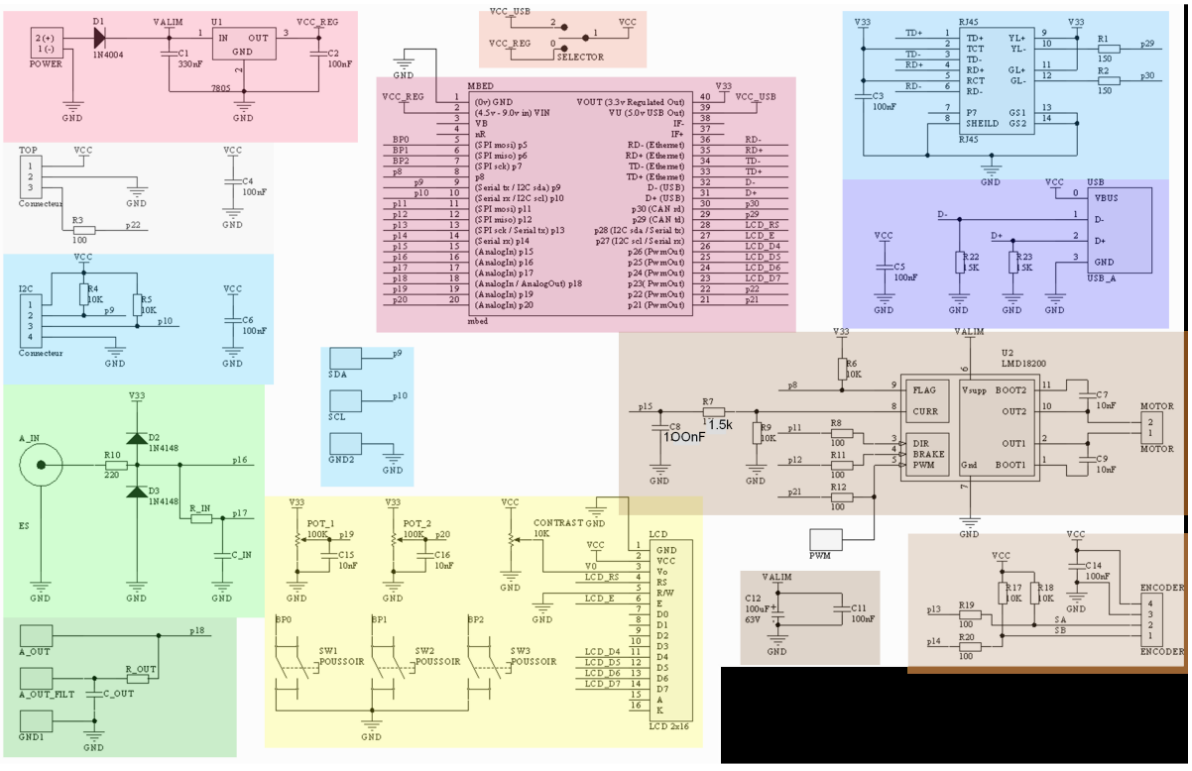


Figure 2: MBED schematics

Benefits and Limitations of using MBED:

Benefits:

- MBED (microcontroller, website and documentation) is user-friendly. The MBED module is really interesting to make practices and projects with students.
- Using well tested libraries is a good practice.
- Use USB link as serial communication with a PC.

Limitations:

- Only features provided by libraries avoid low level programming.
- MBED libraries are not open source not easy to extend low-level implementation of existing libraries and it's difficult to understand the low-level behavior.
- Not yet debugging functionality.

Procedure & Discussion:

Experiment 4 Tasks:

To Do:

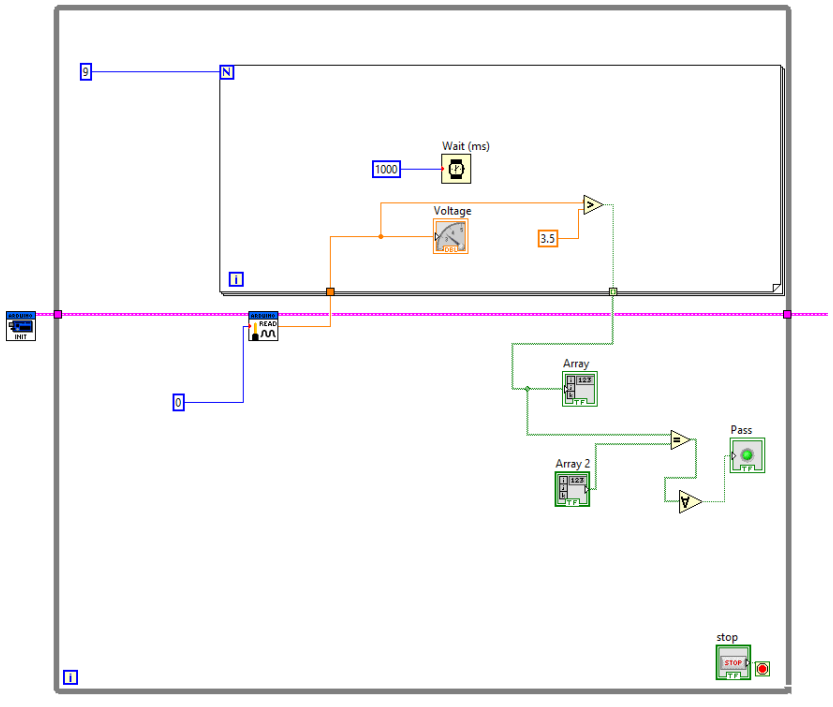


Figure 3: To Do VI Block Diagram

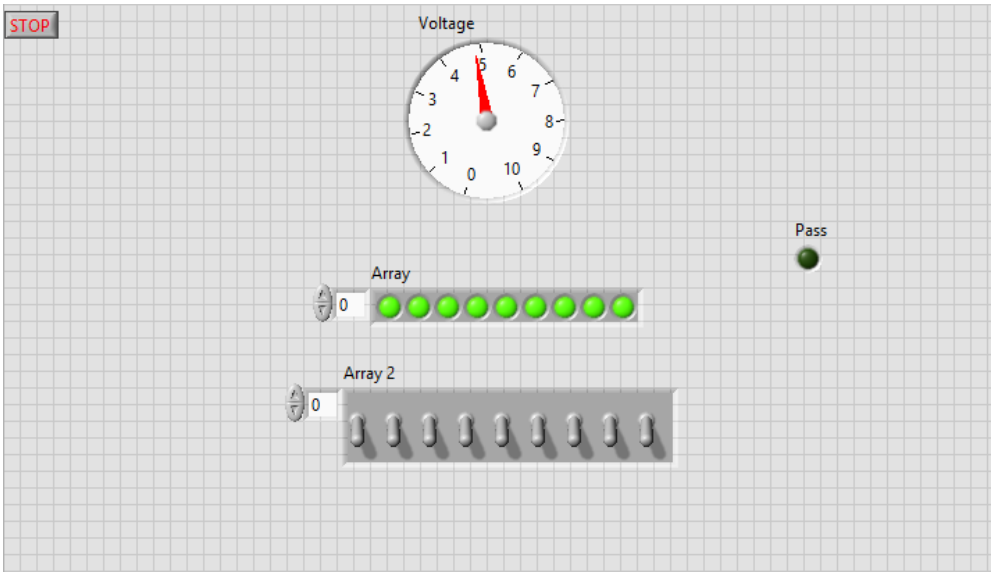


Figure 4: To Do VI Front Panel

In this to do, we have built a 9-bits light detection security key using a LDR with Arduino and LabVIEW, where the user will insert the password using the LDR, if it detects a light it will be a one, otherwise it will be zero. One bit will be read every second. If the password is correct, then the “pass” light will turn on, else, it will stay off.

The correct password was stored in “Array2” (see the VI block diagram figure), and the other array “Array” was used to store the password received from the user. A comparator, will compare each bit in the first array with each corresponding bit in the second array, and produce an array of ones and zeros, depending on the comparison result. If all bits in the resultant array (resultant from the comparison) were ones, then the entered password match the stored one, else it doesn’t, so we need an AND array, so that it make an ANDing operation between the bits of the resultant array so that we have one single bit that determines if the passwords match or not, this bit will be one if they are match, and zero otherwise, and so we connected it to the “pass” LED, so that it will turn on when the correct password is entered.

Experiment 5 Tasks:

Processing "Hello, world!":

```
import processing.serial.*;

void setup(){
  println("Hello, World!");
}

void draw(){
}
```

As simple as that, this will only print "Hello, World!".

Sketch Rectangle, Ellipse, and Line:

```
import processing.serial.*;

PImage image;
void setup(){

  size(1580, 572);                              // The size of the image used
  image = loadImage("image.PNG");
  background(image);                        // set the loaded image as background
}

void draw(){

  ellipse(100, 100, 80, 150);// position = (x,y) = (100,100), dimensions = 80*150
  rect(300, 100, 100, 120);  // position = (x,y) = (300,100), dimensions = 100*120
  stroke(200);
  line(400, 400, 600, 700);  // from (x1,y1) = (100,100) to (x2,y2) = (600,700)
}
```

In this part, we are just drawing an ellipse, rectangle and a line, on a window with a specific background that was chosen by us. The ellipse will be drawn at the point (x,y) = (100,100), with dimension of 80*150, and the rectangle will be drawn at the point (x,y) = (300,100), with dimensions of 100*120. The line will be drawn from point (100,100) to (600,700). See the following figure for outputs.

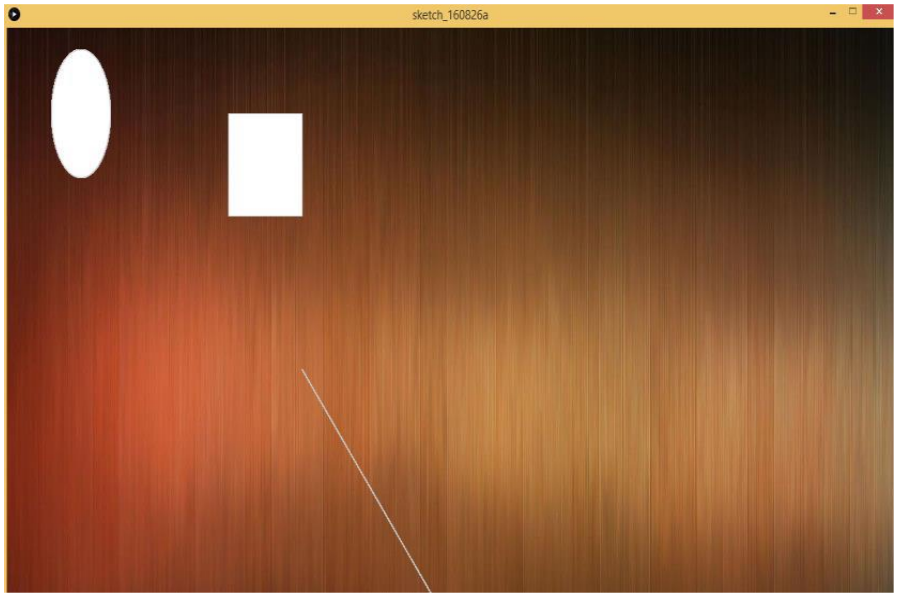


Figure 5

Animation:

```
float x = 50;
float xDelta = 1;
float y = 50;

void setup(){
  /*Frame size*/
  size(500, 100);

  /* For animation, frame rate */
  frameRate(60);

}

void draw (){
  background(0);          // Black background color
  ellipse(x, y, 25, 25); // Ellipse shape
  fill(255, 0, 0);        // RGB color, Red color

  /* External disk, green disk */
  stroke(0, 255, 0);      // RGB, Green color
  strokeWeight(8);        // Width of the green color disk

  x = x + xDelta;// add xDelta to x each iteration

  /* Control direction and move of the shape, frame size is 500 */
  if (x > 500)
    xDelta = -1;

  if (x < 0)
    xDelta = 1;
}
```

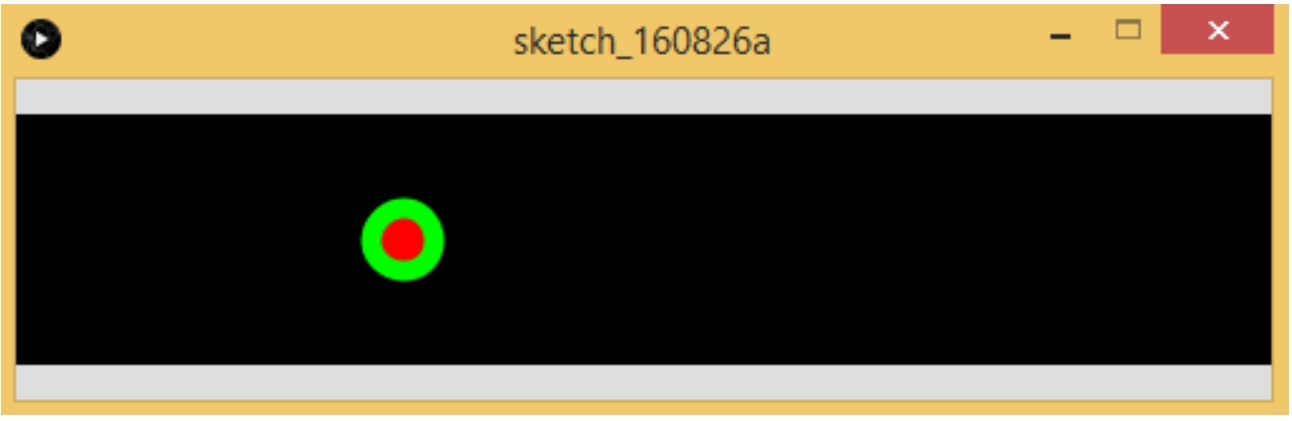


Figure 6

This code will generate a black background, and draw a circle, just like the one shown in the previous figure. This circle will keep moving from left to right and from right to left, it will start from the point (x,y) = (50,50), the y-coordinate will be kept fixed, but the x-coordinate will keep changing, xDelta = 1, will be added to x in each iteration, until x > 500, i.e. the circle reached the right edge of the window, when this happen, xDelta will be changed from +1 to -1, so that -1 will be added now to x, in each iteration, and so the circle will move from right to left, until it reaches the left edge, and then xDelta will become +1 again.

Send data from Arduino to Processing over the serial port:

Arduino Code:

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

}

void loop() {
  Serial.println("Hello, world!");
  delay(100);
}
```

This is Arduino part of code, it will send data from the Arduino to the processing program in PC.

Processing Code:

```
import processing.serial.*;
Serial myPort;
String val;

void setup(){
  String portName = "COM10";
  myPort = new Serial(this, portName, 9600);
}

void draw(){

  if(myPort.available() > 0)              // check if data is available
    val = myPort.readStringUntil('\n');   // read data until new line

  if (val != null)     // if data read is not null, print it
    print(val);

}
```

This code will simply receive the data from the serial port and print it to the screen.

Send data from Processing to Arduino over the serial port:

Arduino Code:

```
int val;
int ledPin = 11;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {

  if(Serial.available()){

    val = Serial.read();
    if(val == '1')
      digitalWrite(ledPin, HIGH);
    else
      digitalWrite(ledPin, LOW);

    delay(10);
  }
}
```

This code will simply, check for availability of data in the serial port, and if the data is available, it will read it. If the read data was ‘1’, then it will turn the LED on, otherwise, it will turn it off.

Processing Code:

```
import processing.serial.*;
Serial myPort;
String val;

void setup(){
  size(200, 200);
  String portName = "COM10";
  myPort = new Serial(this, portName, 9600);
}

void draw(){

  if(mousePressed == true)
    myPort.write('1');
  else
    myPort.write('0');
}
```

This code will create a window if size 200*200, so that if we click on it, it will send ‘1’ to the serial port, otherwise it will keep sending ‘0’.

Pong Game:

Arduino Code:

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

void loop() {

  // Read the inputs
  int leftPaddleUp = analogRead(A0);
  int leftPaddleDown = analogRead(A1);
  int RightPaddleUp = analogRead(A2);
  int RightPaddleDown = analogRead(A3);

  Serial.print(leftPaddleUp);
  Serial.print(",");
  Serial.print(leftPaddleDown);
  Serial.print(",");
  Serial.print(RightPaddleUp);
  Serial.print(",");
  Serial.print(RightPaddleDown);
  delay(10);

}
```

For our Game we have four inputs (two photocells per player).The Controller code works both with the one player mode and two player mode of the pong game, and it is fairly straightforward. The code reads the values which are output by the photocells and sends each value by serial communication to the processing engine.

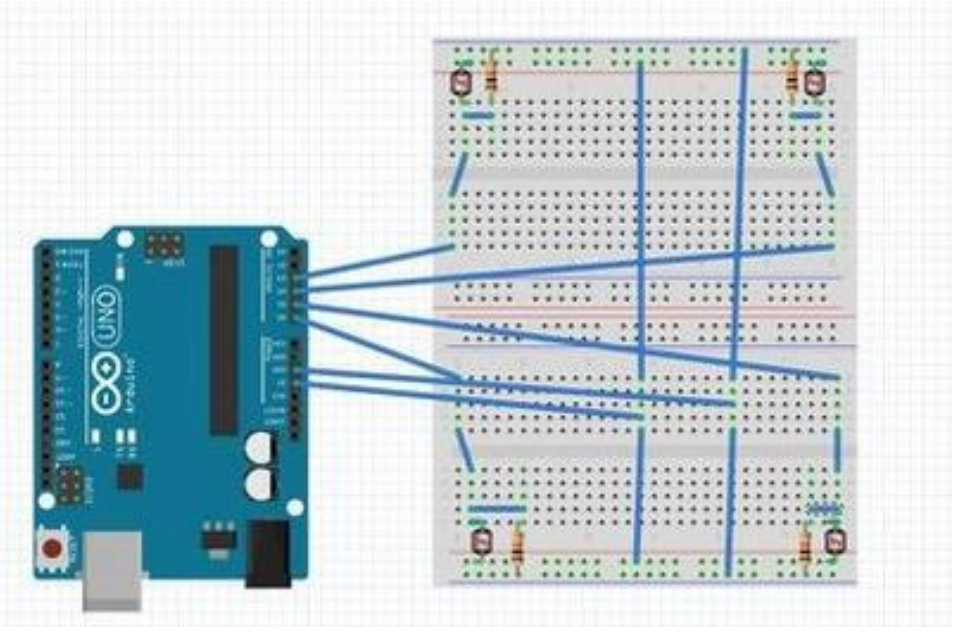


Figure 7: Arduino Connections

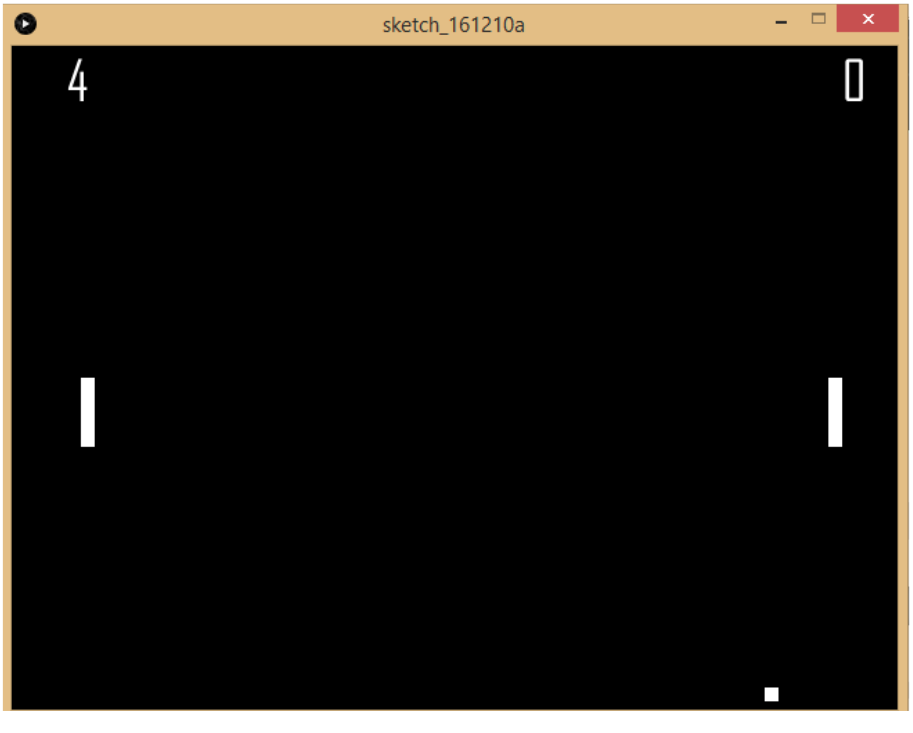


Figure 8: Gameplay

You can play with one player by open Processing and type Player1.txt file instead of Player2.txt file.

Experiment 6 Tasks:

Create your first program:

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {

      while(1){
            myled=0;
            wait(0.2);
            myled=1;
            wait(0.2);
      }
      return 0;
}
```

The first code was very simple, it just aims to make a LED flashing every 0.2 seconds.

Now, to make the four LEDs flash sequentially (i.e. LED1 flashes, then LED2 ,then LED3, then LED4), we just need to define four LEDs as digital output, and repeat the previous code four times in the while loop, just like this code:

```
#include "mbed.h"

DigitalOut myled_1(LED1);
DigitalOut myled_2(LED2);
DigitalOut myled_3(LED3);
DigitalOut myled_4(LED4);

int main() {

      double delay = 0.5;

      while(1){

      myled_1=0;
      myled_2=0;
      myled_3=0;
      myled_4=0;

      myled_1=1;
      wait(delay);
      myled_1=0;
      wait(delay);

            myled_2=1;
            wait(delay);
            myled_2=0;
            wait(delay);

            myled_3=1;
            wait(delay);
            myled_3=0;
            wait(delay);

            myled_4=1;
            wait(delay);
            myled_4=0;
            wait(delay);
      }
}
```

Serial communication via USB:

```
#include "mbed.h"
#include "TextLCD.h"

Serial pc(USBTX, USBRX); // Tx, Rx

int main() {

      int i = 0;

      pc.baud(115200);
      pc.printf("MBED\r\n");

      while(1){
            pc.printf("%d\r\n", i);
            i++;
            wait(1);
      }

}
```

This simple code will run on the MBED, and send data through the serial cable (USB) to the computer with a baud rate of 115200. The data will be “MBED” word, followed by a counter that counts from zero and keep counting in an infinite loop. We can observe the incoming data to the computer using serial monitor, as shown in the following figure:

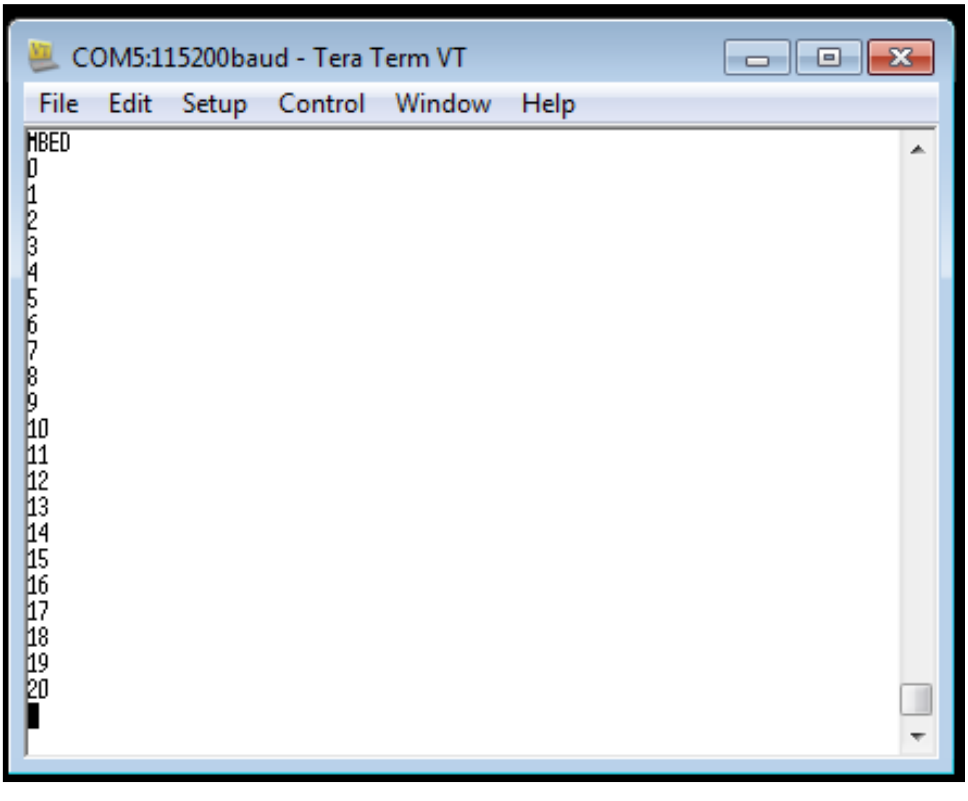


Figure 9: Incoming data from MBED to PC

TextLCD:

```
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p28, p27, p26, p25, p24, p23);// rs, e, d4-d7

int main() {
      lcd.printf("Hello World!\n");
      return 0;
}
```

As simple as that ! This code will print “Hello World!” on the screen ! See the following figure:

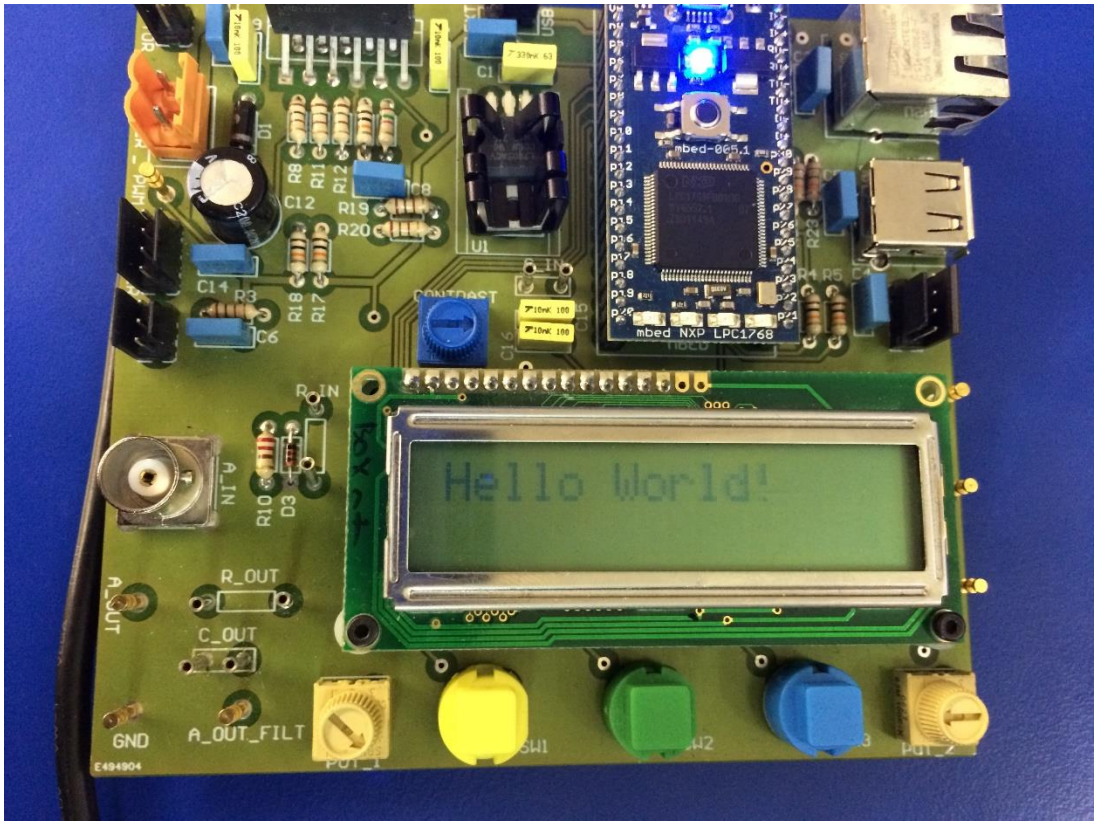


Figure 10: LCD displaying the sent data

To Do:

```
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p28, p27, p26, p25, p24, p23);// rs, e, d4-d7
Serial pc(USBTX, USBRX);

bool flag = true;

int main() {

       pc.baud(115200);
       int i = 0;

       while(1){
                  if ( pc.readable() ){

                        char c = pc.getc();
                        if (c == '1')
                              lcd.printf("11111111111111\n");

                        if (c == '0')
                              lcd.printf("00000000000000\n");
                  }

                  wait(1);
      }
}
```

This time, we are sending data from Computer to MBED. We will send either 0 or 1, the code in the MBED will first check if there is a character to read from PC (pc.readable), and if there is, it will read it using the function (getc), and print ones if the received character is ‘1’, or zeros, if the received character is ‘0’.

Conclusion:

From these experiments, students are supposed to be more familiar with LabVIEW and its applications with Arduino. Also, they should have had a good overview on the Processing Language and the MBED microcontroller. We saw how processing language can be used for learning, prototyping, and production. It also can be used for interactive programs with 2D, 3D, or PDF output. On the other hand, MBED has had a good introduction that is enough to show how much it is user-friendly and simple to use and how interesting it is for making practices and projects with students.

To conclude, these experiments introduced us to many software and hardware resources that can be very useful for technical projects and studies.

References:

[1]: Interfacing Lab Manual - Experiment 3 (Accessed: 11 April 2017).

[2]: Interfacing Lab Manual - Experiment 5 (Accessed: 11 April 2017).

[3]: Interfacing Lab Manual - Experiment 6 (Accessed: 11 April 2017).