Department of Computer Engineering
Faculty of Engineering
University of Jaffna
EC2010 – Computer Programming
Lab 07

Microcontroller Programming with Arduino

Date: 16.01.2023 - 19.01.2023 **Duration: 3 hours**

Objectives: Understand the basics of Microcontroller programming.
Write programs in C for Arduino Microcontroller.

Submission instructions:

- Any plagiarized work will be given 0 marks.
- Submit your lab work as a zip file named LAB06_20YYEXXX (20YYEXXX – Your Registration Number)before the given deadline via teams.
- Your lab report should have a cover page with essential details.
- Answers to each question should have a code screenshot, code in copyable text format, a picture of your design, and the corresponding output screenshot.
- 85% of marks for this lab will be evaluated from lab experiments. And 15% will be from submitted report.

Introduction:

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

When you load the Arduino program you are greeted by the following simple IDE:

1. **Verify** **:** Compiles and checks your code. It will catch errors in syntax.
2. **Upload** **:** Sends your code to the Arduino board. When you click it, you should see the lights on yourboard blink rapidly.
3. **New** **:** This button opens a new code window tab.
4. **Open** **:** This button will let you open up an existing program.
5. **Save** **:** This saves the currently active program.
6. **Serial Monitor** **:**This will open a window that displays any serial information your Arduino boardis transmitting. It is very useful for debugging.

7. **Program Name** :This shows the name of the program you are currently working on.

8. **Code Area** :This is the area where you write the code for your program.

9. **Message Area** :This is where the IDE tells you if there were any errors in your code.

10. **Text Console** :The text console shows complete error messages. When debugging, the text console isvery useful.

11. **Board and Serial Port** : Shows you what board and the serial port selection.

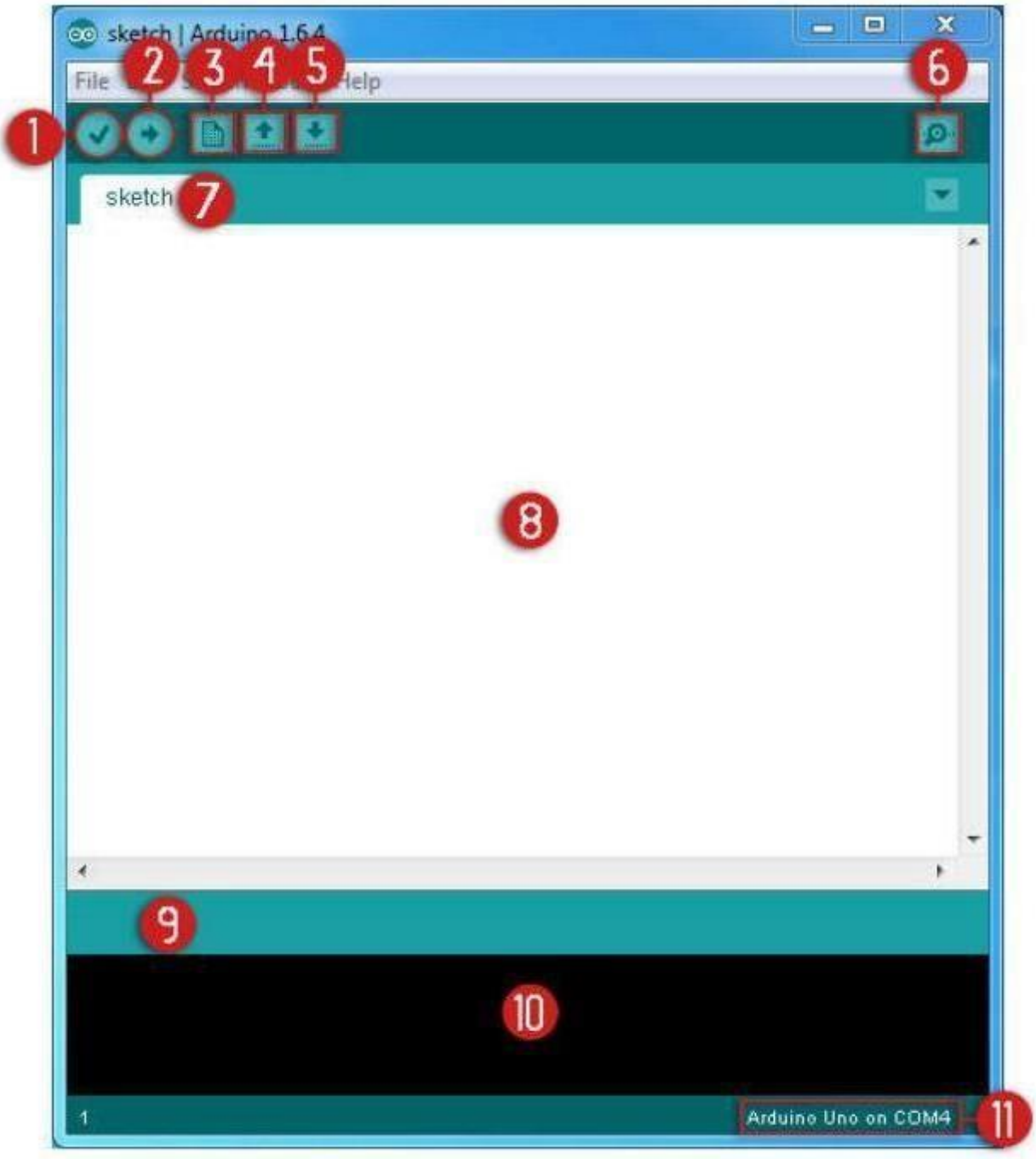


Figure 1: Arduino IDE

1. Blinking LED (Digital output)

The blinking LED is like the "Hello World!" of physical computing.

The first thing we do is define a variable that will hold the number of the pin that the LED is connected to. We don't have to do this (we could just use the PIN throughout the code) but it makes it easier to change to a different pin. We use an integer variable (called an int).

```
int ledPin = 13;
```

The second thing we need to do is configure as an output the pin connected to the LED. We do this with a call to the pinMode() function, inside of the program's setup() function:

```
void setup()

{
        pinMode(ledPin, OUTPUT);
}
```

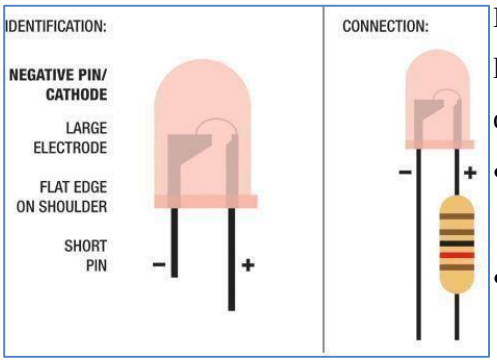
Finally, we have to turn the LED on and off with the program's loop() function. We do this with two calls to the digitalWrite() function, one with HIGH to turn the LED on and one with LOW to turn the LED off. If we simply alternated calls to these two functions, the LED would turn on and off too quickly for us to see, so we add two calls to delay() to slow things down. The delay function works with milliseconds, so we pass it 1000 to pause for a second.

```
void loop()
{
        digitalWrite(ledPin, HIGH);delay(1000); digitalWrite(ledPin, LOW);delay(1000);
}
```

Full program code: Blinking LED

```
int ledPin = 13;   // LED connected to digital pin 13

void setup()
{
        pinMode(ledPin, OUTPUT);       // sets the digital pin as output.
}

void loop()
{
        digitalWrite(ledPin, HIGH);             // sets the LED on
        delay(1000);                            // waits for a second.
        digitalWrite(ledPin, LOW);              // sets the LED off.
        delay(1000);                            // waits for a second.
}
```

2. Connecting an LED



LEDs have polarity, which means they will only light up if you orient the legs properly. The long leg is typically positive and should connect to a digital pin on the Arduino board. The short leg goes to GND.

- To protect the LED, you will also need to use a resistor "in series" with the LED.
- If the LED doesn't light up, try to reverse the legs (you won't hurt the LED if you plug it in backward a shortperiod

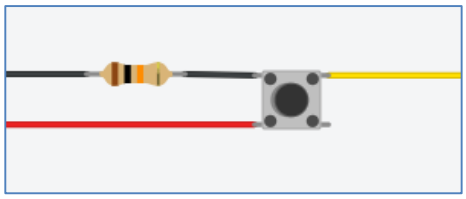
Exercise 1: **Connect 2 LEDs on Digital pin 12, 11 and program to blink them alternatively.**

Exercise 2: **Find the code below for the knight rider circuit. Reproduce the code in your editor and try to understand how this works. You will be evaluated based on your understanding.**

Full program code: Knight Rider

```
void setup()
{
        for(int i=2; i<7; i++)
        {
                pinMode(i,  OUTPUT);
        }

}

void loop()
{
        for(int i=2; i<7; i++)
        {
                digitalWrite(i, HIGH);
                delay(20);
                digitalWrite(i+1,   HIGH);
                delay(20);
                digitalWrite(i+2,   HIGH);
                delay(20);
                digitalWrite(i, LOW);
                delay(20);
                digitalWrite(i+1,   LOW);
        }

        for(int i=7; i>2; j--)
        {
                digitalWrite(i, HIGH);
                delay(20);
                digitalWrite(i-1,   HIGH);
                delay(20);
                digitalWrite(i-2,   HIGH);
                delay(20);
                digitalWrite(i, LOW);
                delay(20);
                digitalWrite(i-1,   LOW);
        }
}
```

3. Pushbutton (Read Digital Values)



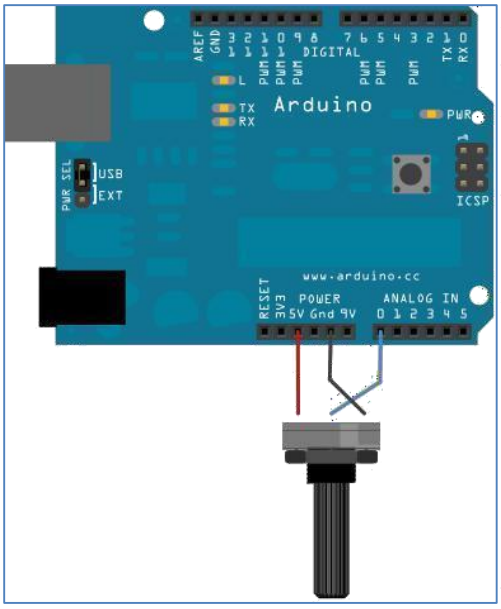
The pushbutton is a component that connects two points in a circuit when you press it. The example turns on an LED when you press the button. We connect three wires to the Arduino board. The first goes from one leg of the pushbutton through a pull-up resistor to the 5-volt supply. The second goes from the corresponding leg of the pushbutton to ground. The third connects to a digital I/O pin (here pin 7) which reads the button's state. When the pushbutton is open (un-pressed) there is no connection between the two legs of the pushbutton, so the pin is connected to 5 volts (through the pull-up resistor) and we read a HIGH. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to ground, so that we read a LOW. (The pin is still connected to 5 volts, but the resistor in-between those means that the pin is "closer"to ground.)

Full program code: Program to on-off led using a push button as a switch.

```
int ledPin = 13;                                    // choose the pin for the LED
int inPin = 7;                                      // choose the input pin (for a pushbutton)
int val = 0;                                        // variable for reading the pin status

void setup()
{
      pinMode(ledPin, OUTPUT);                      // declare LED as output.
      pinMode(inPin, INPUT);                        // declare pushbutton as input.
}

void loop()
{
      val = digitalRead(inPin);                     // read input value

      /* check if the input is HIGH (button released)*/

      if (val == HIGH){
            digitalWrite(ledPin, LOW);              // turn LED OFF
      }else{
            digitalWrite(ledPin, HIGH);             // turn LED ON
      }
}
```

Exercise 3: Program to on-off led using a push button as a switch.

4. Reading a Potentiometer (Analog reading)



A potentiometer is a simple knob that provides a variable resistance, which we can read into the Arduino board as an analog value. In this example, that value controls the rate at which an LED blinks.

We connect three wires to the Arduino board. The first goes to ground from one of the outer pins of the potentiometer. The second goes from 5 volts to the other outer pin of the potentiometer. The third goes from analog input 2 to the middle pin of the potentiometer.

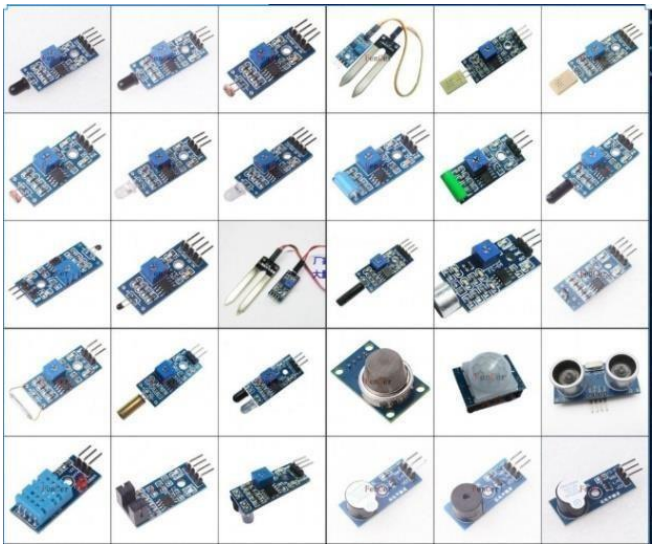
By turning the shaft of the potentiometer, we change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer. This changes the relative "closeness" that pin to 5 volts and ground, giving us a different analog input. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and we read 0. When the shaft is turned all the way in the other direction, there are 5 voltsgoing to the pin and we read 1023. In between, analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

Exercise 4: Write a program to check the values of potentiometer and display it in the serial monitor.

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to

int sensorValue = 0;  // value read from the pot

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);

}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");

  Serial.print(sensorValue);

  Serial.print("\n");

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(200);
}
```

5. Sensors

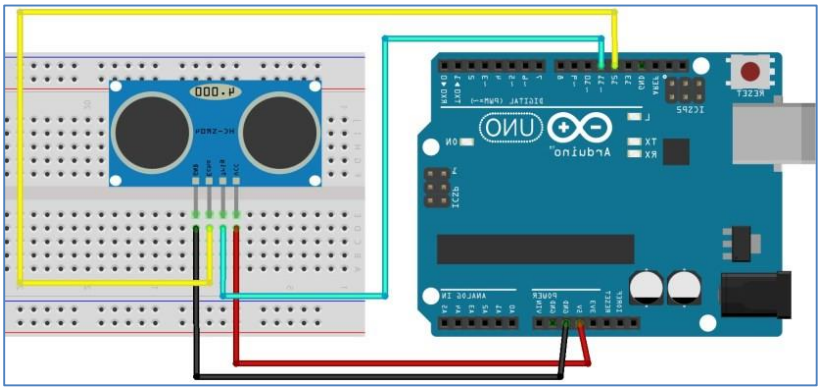
Sensor is a device, module, machine, or subsystem whose purpose is to detect events or changes in its environment and send the information to the electronics, frequently a processing unit.



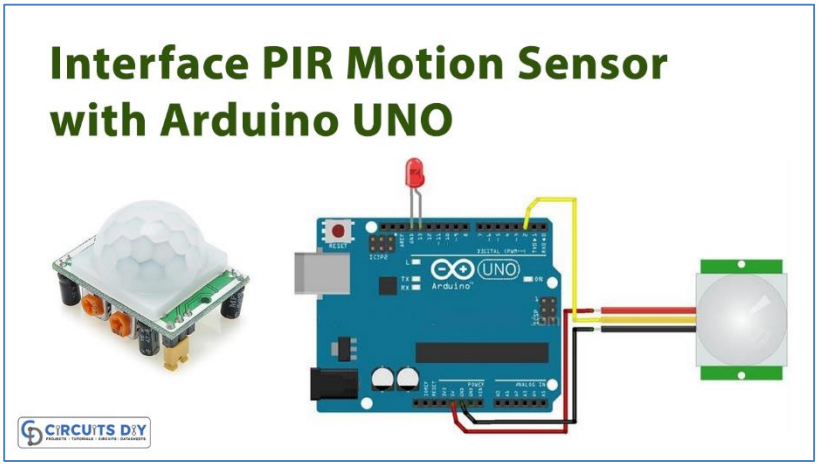
Different types of Sensors

- Temperature Sensor
- Proximity Sensor
- Accelerometer
- IR Sensor (Infrared Sensor)
- Pressure Sensor
- Light Sensor
- Ultrasonic Sensor
- Smoke, Gas and Alcohol Sensor
- Touch Sensor
- Color Sensor
- Humidity Sensor
- Flow and Level Sensor

Exercise 5:Design an Arduino circuit to measure the distance between the ultrasonic sensor and any object. Then program it to light an LED when the distance between sensor and object below 30cm.



Exercise 6: Design an Arduino circuit to Sense motion using PIR Sensor and light up an LED when motion detected.

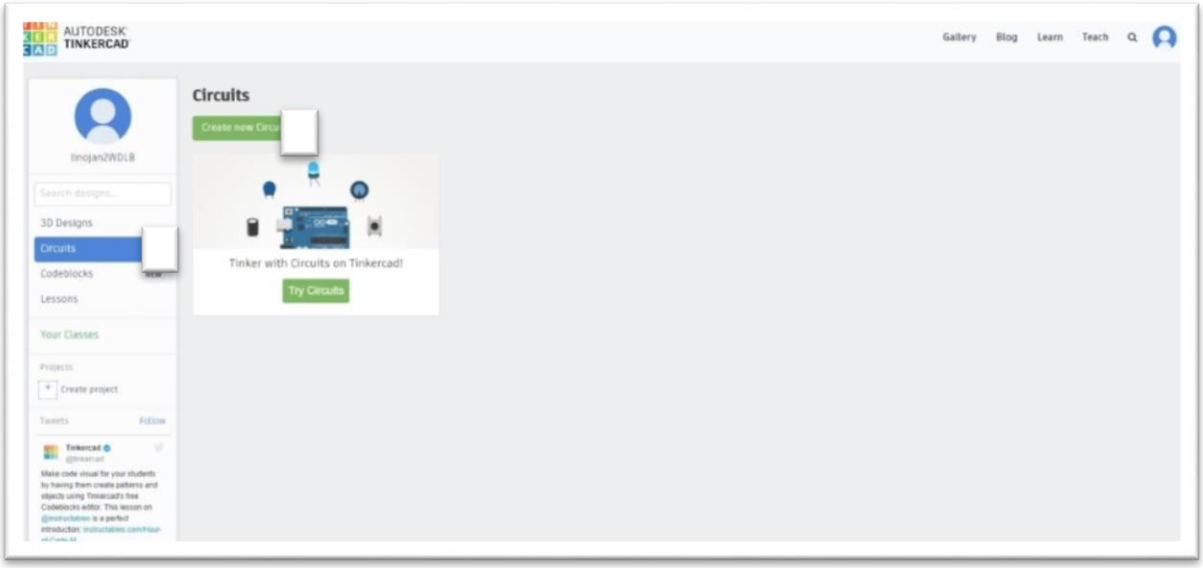


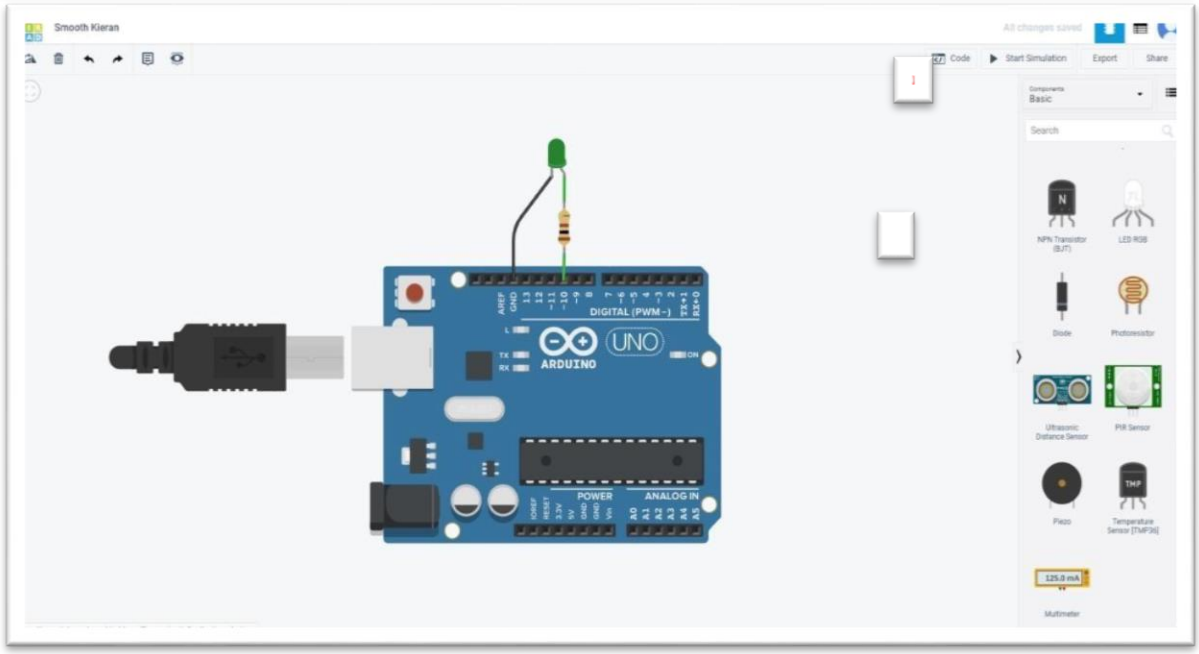
Annexed

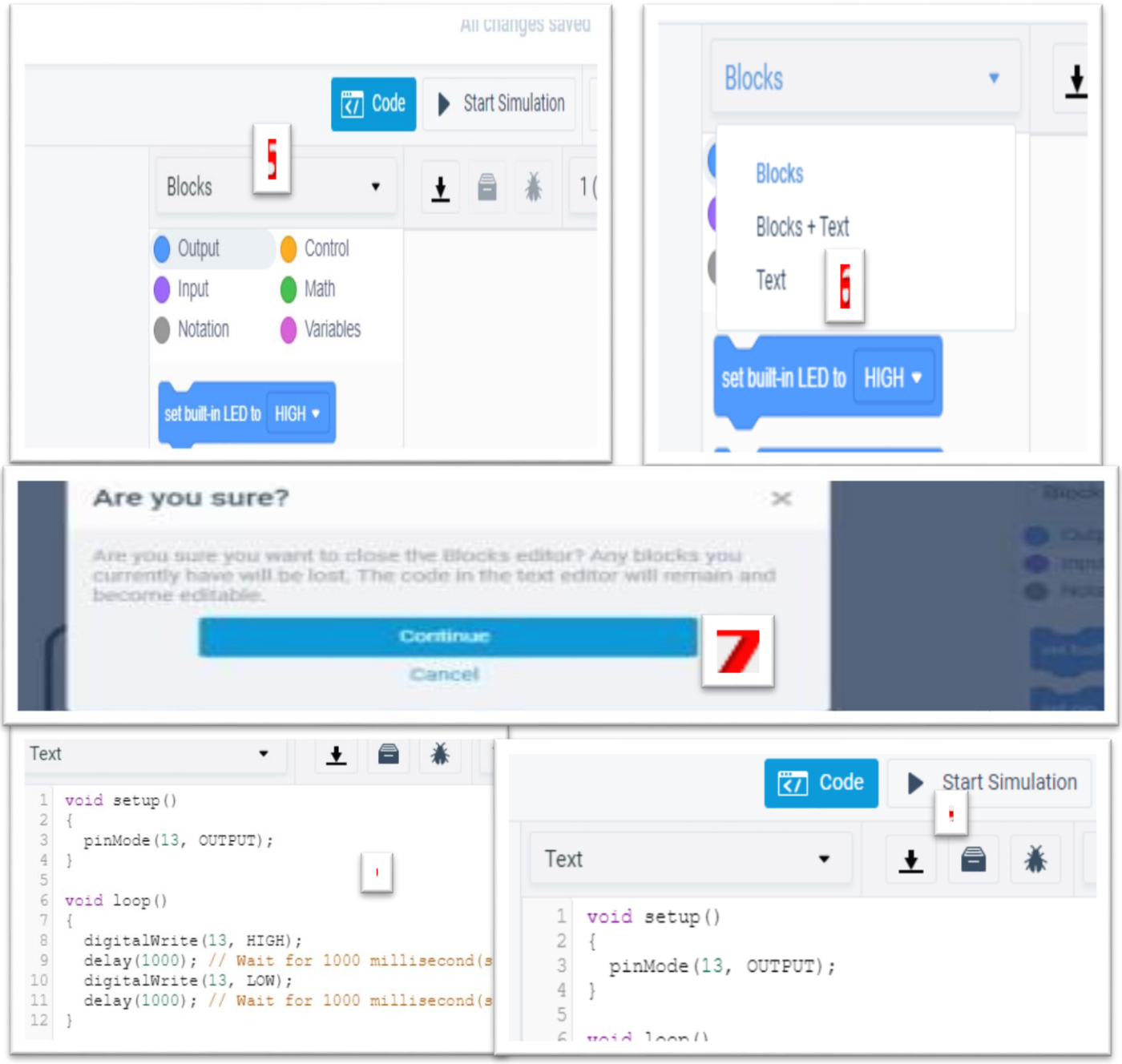
Simulation

TINKER CAD – Online Simulator (Link: https://www.tinkercad.com/)

1. By clicking the Link**:** https://www.tinkercad.com/ you can create an account using themail ID providedby the faculty.
2. After creating the account, by following 9 steps you can easily simulate Arduino circuit.







1. In your account click **circuit**
2. Create a **new Circuit**
3. Through drag and drop design your circuit
4. Click **Code**
5. Click **Block**
6. Select **Text**
7. Click **Continue**
8. Write your code
9. **Start Simulation**