Robotics II: Introduction to programming

Note: We will be completing this lab in alternate teams of 2 to get more hands-on experience with programming and circuit-building! Practice new skills and come back next time to bring skills to original team :)

The deliverable from this lab is the commented code and a video of each circuit uploaded to your team webpage. Make sure to save all code, comment, and take a video of your circuit operating throughout the lab!

Brief Background

All engineers require some knowledge of electronic components and basic programming. A basic understanding of how electronic circuits and devices (such as microcontrollers) impact our daily lives serves to enhance our knowledge of engineering. Today's lab is the continuation of our exploration of electronic circuits and programming towards our final design robot project!

Objectives

The objective of this lab is to become introduced to programming. In completing this lab you will:

- Learn to plan programs using flowcharts and pseudocode
- Employ basic structures and functions to write programs
- Run circuits containing LEDs and switches
- Communicate your results to the class

Activity 1 – Simple programming using Arduino (35 min time limit!)

Using TinkerCAD, build a single LED circuit (include the series resistor!). *Note: you can use your LED circuit from the Week 6 Circuits Lab if you saved it!* Instead of using a battery for the power source, select an Arduino Uno from the components list and connect the circuit to I/O pin 2, and the 'ground' power rail to the GND rail, as shown in Figure 1.

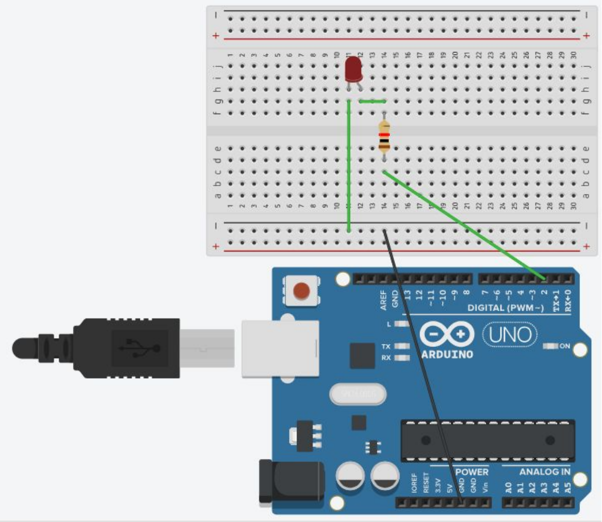
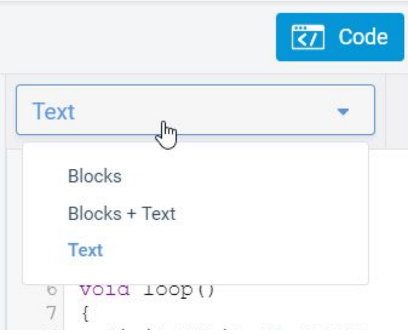


Figure 1: TinkerCAD simulation with Arduino Uno of LED circuit

Next, open the 'Code' window by pressing the button on the top menu bar, and select 'Text' from the drop down menu (shown in Figure 2). The default code that is shown in the window is very close to the actual code you need to run this example! Change the pin number and delay values so that your LED will flash at a frequency of 2 Hz.



```
void loop()
{
```

Figure 2: Code window in TinkerCAD

Now, demonstrate that your team can build a circuit and use the Arduino to control the lighting of a LED by making a LED flasher using the grading criteria in Table 1! *You can continue using the simulation to help you code the LEDs, but this is now optional.* When finished, ensure your code is well commented, save for later use, then take a video of your working circuit.

Table 1: LED Flasher grading criteria

Points	Description
0	You do not do the assignment (or do not participate within your team)
1	An LED flasher that does not work at all
2	An LED flasher that works but does not meet any of the specifications below
3	Flashes one LED at **exactly 2 Hz**
4	An interesting flasher that flashes two or more LEDs (one must continue flashing at 2 Hz, the other must be at a different frequency)
5	A truly remarkable LED flasher that amazes and delights (minimum 3 LEDs with only one still flashing at exactly 2 Hz - e.g. three unique frequencies)

Activity 2 – Using variables and logic constructs

Logic constructs require the use of relational operators to determine which branch or loop of the program to execute. Example relational operators are shown for Arduino code in Table 2. Examples of each logic construct and example code are in Table 3. Use these logic constructs in conjunction with variables to complete the following exercise.

Exercise: Write a program that turns on an LED attached to Pin3 for 5 seconds then turns it off. The light should stay off for 3 seconds and then the LED should blink 10 times at a rate of 1 Hz (on and off each second).

a. Build and run your program to test it! Take video of function.
b. Comment your code and save.
c. Upload to the website when complete.

Table 2: Example Arduino code relational operators (notes that you need 2 equal signs if 'equal to')

operator	==	!=	<	<=	>	>=
meaning	equal to	not equal to	less than	less than or equal to	greater than	greater than or equal to

Table 3: Example Arduino code for logic constructs

Selection (if; if-else)	**Example Selection Code**

if (expression)
{
   Statements //executed if
              //expression is true
}
	if (x !=0)
	{
	  z=y/x; //allows division only
	          //when x is not zero!
	}

if (expression)
{
   Statements1 //executed if
               //expression is true
}
else
{
   Statements2 //executed if
               //expression is false
	if (x>y)
	{
	  z=z+x; //adds larger of x
	          //and y to z
	}
	else
	{
	  z=z+y;
	}

Loops (for; while)	**Example Loops Code**

for (initialization;expression;increment)
               //if expression is true;
               //re-evaluated after all
               //statements are executed -
               //exits when false
{
  Statements   //executed if
               //expression is true
}
	int i;
	for(int i=0;i<10;i++)
	{
	  digitalWrite(ledPin,HIGH);
	  delay(500);
	  digitalWrite(ledPin,LOW);
	  delay(500);
	}

while (expression)
               //if expression is true;
               //re-evaluated after all
               //statements are executed -
               //exits when false
{
  Statements   //executed if
               //expression is true
}
	var=0;
	while(var<200)
	{
	   var++; //same as var=var+1;
	}

Activity 3 – Using buttons with Arduino

Push buttons were introduced during the last lab as a switch for controlling current. Now we will investigate another use of the button/switch which is to **provide input** to the Arduino. The microcontroller waits for the circuit to change (button to be pressed) then can take action based on the press.

Before we can introduce code to signify a button press, we must first understand how to wire a button to the Arduino so it can "see" the input signal.. The button can be wired in two states - 'active high' or 'active low'. 'Active high' means that when pressed, the button sends a 'high' signal to the Arduino (a ‘1’ value); ‘active low’ sends a ‘low’ signal (a ‘0’ value) to the Arduino when pressed. The two wiring schematics are shown below in Figure 3.

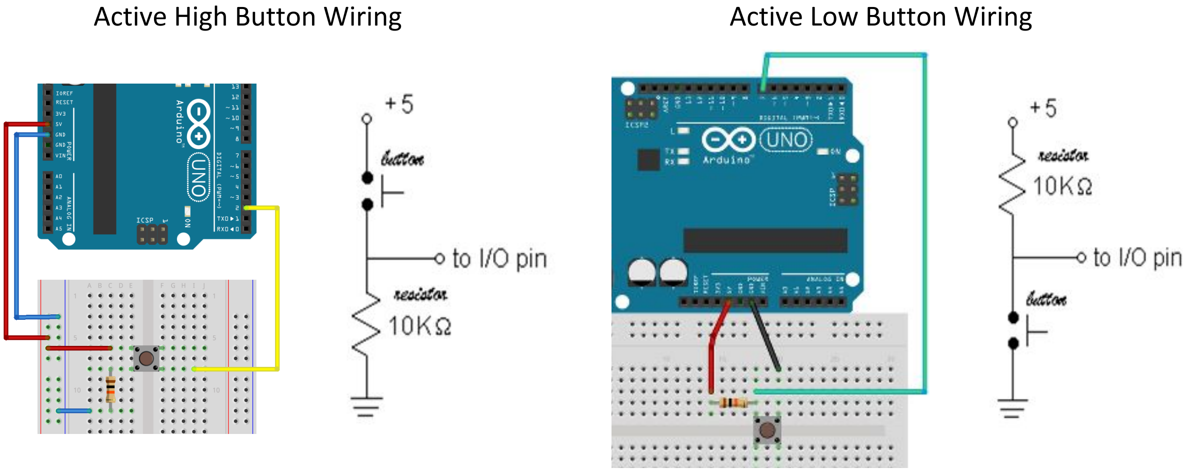


Figure 3: Push button wiring and code

With any button wiring, the code to provide input to the Arduino is the same `digitalRead()` function. **Example code is available in the Appendix.**

Push button exercises - For each exercise below (A-E), write a sketch, build the circuit and test functionality. Comment all code and save making sure each program is clearly labeled and document your circuit by drawing an electronic schematic for each. Show your TA exercises A-C when circuit is working, along with its schematic for credit! Exercises D & E will be uploaded to your team’s webpage.

A. Write a sketch that turns on an LED for 3 seconds every time a button is pressed.

B. Write a sketch that flashes an LED 10 times at a rate of 2 Hz every time a button is pressed. When button not pressed, LED should be off.

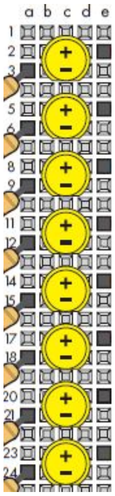
C. Write a sketch that starts by having an LED go on. The LED will stay on until a button is pressed. A half second after the button is pressed, the LED will go off and an alternate color LED will go on for 5 seconds. After this the whole cycle repeats.

D. Write a sketch that starts out by flashing an LED. On a button press, the LED turns on for 2 seconds. After this the whole cycle repeats.

E. Write a sketch that lights an LED. The LED should turn off after a button has been pushed 3 times. Assume there is least 1/4 second between button presses.

Activity 3A – Extra Points Practice Exercises - Complete if time permits! (*Optional* - *Individuals can complete; does not have to be team effort*)

Check your understanding of LED circuits and coding by completing the following exercises. You may check out the needed supplies from the ENGR 1 lab supplies - just ask! To turn in, each exercise completed should be uploaded to the webpage (***commented*** code and invites or videos), with clear ownership (name(s)!) of who completed this portion of the lab.

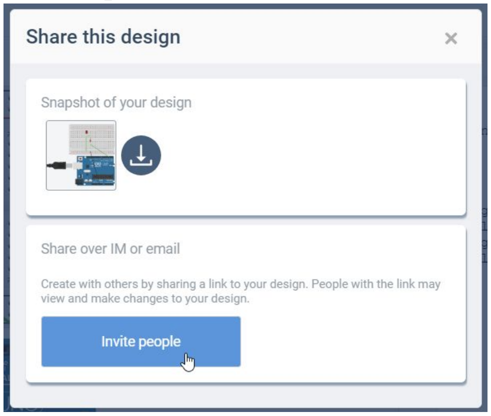


Setup: set up eight LEDs, each on its own circuit. Ensure that the LEDs are arranged in a single line, as shown on the right. Keep your circuits neat and tidy so you can easily follow the connections!

Initial Test: Code a program **using arrays** to have all LEDs blink at once at a frequency of 1 Hz. This is done to verify that the LEDs are all correctly connected and pins are all working.

EC1. (10 pts *possible*) "One after another" – Write a program which lights all the LEDs in turn (light one LED, then the next (keeping the first on), then the next (in order), until all LEDs are lit), then turns off the LEDs in the reverse order. Have the program loop indefinitely. Continue to use your arrays!

- (5 pts) Use TinkerCAD, build the circuit and use the code editor to input your program. When finished, invite your professor and TA using the 'Share' button on the top menu and 'Share over email'.



Create with others by sharing a link to your design. People with the link may view and make changes to your design.

- (5 pts) Physically build the LED circuits using a breadboard and an Arduino Uno. Take a video of your working circuit! Remember to set permissions properly so your TA and Professor can view!

EC2. (10 pts *possible*) "Ping pong" - Write a program which steps through the LEDs, lighting one at a time (one on, all others off) starting from one end then returning upon reaching the other end. Have each LED light for a period of 100 ms. Again, use arrays in your program. Also, simplify your code by using a variable for the light time of the LED. This simplification means that you only have to change the value once in the program rather than multiple times. Change the light time from 100 ms to 50 ms.

- (5 pts) Using TinkerCAD, build the circuit and the code editor. When finished, invite your professor and TA using the 'Share' button on the top menu and 'Share over email'.
- (5 pts) Build circuit on breadboard, connect to Arduino, and take a video of your working circuit!Remember to set permissions properly so your TA and Professor can view!

Activity 4 – Communication of lab results

Your aim for this deliverable is to prove that your circuits function as intended! Please provide your commented code and a video of circuit operation for:

1. LED Flasher (Activity 1)
2. Logic circuit (Activity 2)
3. Push button exercises (Activity 3 - only exercises D & E, you will show a TA or professor exercises A-C in class)
4. *Extra credit (Activity 3A) - optional*

Your circuit results will be placed on your team's webpage (subpage of our ENGR 1 Programming Lab site). There is an example team submission for your reference, if needed. Place your commented code and a video of your circuit working on your team page for each of the above circuits. Format and style of your page is up to you, but the page should be clear and organized for each circuit (doesn't have to be like the example - just make sure you are clearly communicating the code and function of your circuits! I should not struggle to find information to grade!) One suggestion for including your commented code is to simply take a screenshot of your Arduino IDE with code displayed; just make sure the size is readable on your team page!

Appendix - Example button code using selection logic

```
// constants won’t change. Used here to set pin numbers
const int buttonPin=2;   //set button pin number
const int ledPin=13;     //set LED pin number

//variables will change
int buttonState=0;       //variable for reading pin status

void setup() {
   pinMode(ledPin,OUTPUT);      //declare LED as output
   pinMode(buttonPin,INPUT);    //declare pushbutton as input
   Serial.begin(9600);          //enable serial monitor output
}

void loop() {
   buttonState=digitalRead(buttonPin);  //read input value
   Serial.println(buttonState); //write value to serial monitor to debug

   if (buttonState == HIGH) {           //check if pushbutton is pressed
      digitalWrite(ledPin,HIGH);        //if it is, turn LED on
   }
   else {
      digitalWrite(ledPin,LOW);         //if it isn’t, keep/turn LED off
   }
}
```