Introduction

In this activity, we will cover the basic input and output operations with Arduino IO pins. This will consist of learning how to use the digitalWrite() and digitalRead() functions to interface with outside circuits.

Materials List

1. Arduino
2. USB Cable
3. Bread Board
4. Red LED
5. Pushbutton Switch
6. Toggle Slide Switch
7. 5.1KOhm Resistor x 2 (any resistor from ~1-5K will be compatible)

Controlling Pin13 LED

1. If you look up the schematic for the Arduino Mega [1](Figure 1). You can see there is an LED on the board that is connected to Pin 13.

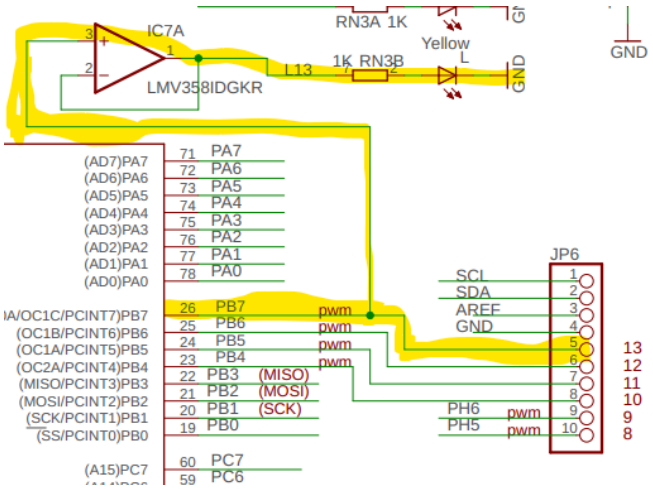

Figure 1: Schematic of onboard LED from the Arduino schematic. IC7, the Op Amp, is acting as a buffer, so the microcontroller does not have to drive the LED.

We need to be careful when we read the Arduino pin numbers off the schematic since the same pin will have multiple numbers. Looking at Figure 1 We see the highlighted wire is labeled 3, 5, 13, and 26. The 26 refers to the pin for the Atmega chip itself and will match the Atmel datasheet. The 3 refers to the Op Amp pin that the digital pin is connected to. The 5 is the numbering on that individual header, so each grouping of 10 pins will be numbered 1-10. The red numbering to the side of each IO header on the drawing corresponds to the labels on the headers themselves.

This is also the numbering that matches how the pins are used in the IDE. In general, we will use this number to refer to the pins. When looking up other documentation be sure you are clear which numbering system it is referring to.

2. We will write a simple program to blink that LED on for 1 second and then off for 1 second. The flow chart for such a program is shown in Figure 2.
3. We will use the **pinMode()**, **digitalWrite()**, and **delay()** functions to accomplish this. Write a program to do this.
4. Once you upload your code, you should see the yellow LED (Not the TX or RX LEDs) blinking on and off.
5. Your code should look similar to that in Figure 3.

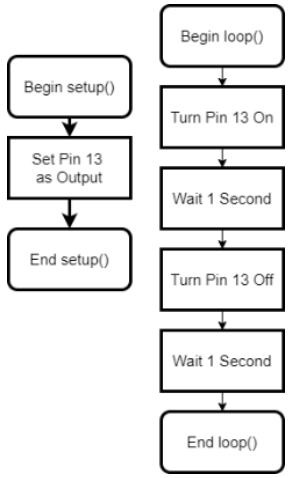

Figure 2: Flowchart for the first simple blink program.

```
1  void setup() {
2    // put your setup code here, to r
3    pinMode(13, OUTPUT);
4  }
5
6  void loop() {
7    // put your main code here, to ru
8    digitalWrite(13, HIGH);
9    delay(1000);
10   digitalWrite(13, LOW);
11   delay(1000);
12 }
13
```

Figure 3: Simple program to blink the internal LED on pin 13.

Controlling an External LED

1. Now, let's interface the Arduino with our breadboard. First, we will use digital pin 4 as a current source by building the circuit in Figure 4
2. For this, we will use the Ardiunos 5V and GND pins for power and ground.
3. Connect the 5V pin of the Arduino to one of the top rows of the breadboard.

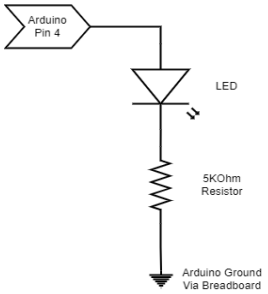

Figure 4: Digital output circuit. Here the pin is supplying current.

4. Connect one of the GND pins of the Arduino to one of the bottom rows. All of the GND pins on the Arduino are connected to each other, so you can use any one of the pins.
5. Now connect pin 4 to one of the columns.
6. Plug the + lead of the LED to that same column on that breadboard. Remember, you can use your multimeter to determine which lead is positive if it has a diode test function. Also, if you look at the LED, one of the leads should be longer than the other; this is the positive lead.

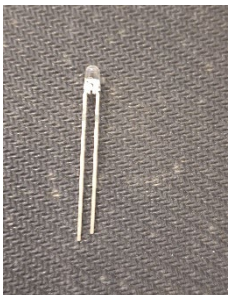

Figure 5: The longer lead of the LED is the positive (anode)

7. Use the LED to bridge the chip gap of the breadboard and plug it into one of the bottom columns. It does not have to be in the same column as the top lead to work, but it is a good habit to keep things vertically lined up.
8. Now use the resistor to connect that column to the ground rail, completing the circuit. When done, it should look like Figure 6.

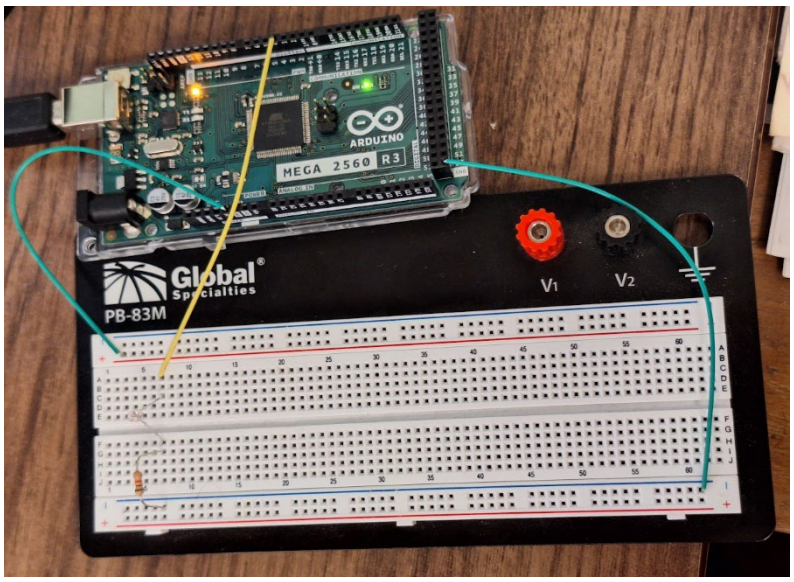

Figure 6: Breadboarded circuit for digital output. The two green jumpers are connecting to 5V and GND on the Arduino, while the yellow jumper connects pin 6 to the positive LED lead.

9. Now we just need to modify the code to change our output pin from pin 13 to pin 4. This requires us to change every time we use 13 to 4. Currently, this is just 3 lines, but as the program becomes more complex, this will become increasingly tedious. A better approach would be to make our output pin a variable assigned in our globals and refer to the variable when we need to access that pin. An example is shown Figure 7. Now we will only have to change the assignment.
10. Modify your code and upload it to the Arduino. You should see the external LED blink.

NOTE: With a 5K resistor, the LED can be somewhat dim from the side. If you look at the LED from above, you should be able to clearly see if it is lit. Swapping out the resistor for a lower value will also make it brighter, but be careful not to exceed the pin's current limit

11. Now modify your circuit so that pin 4 acts as a current sink instead, as shown in Figure 8.
12. Your program should still blink the LED without modification, but now it will be on when Pin 4 is Low. You can confirm this by measuring pin 4's voltage with your multimeter.

```
int outputpin=4;

void setup() {
  // put your setup code here, to run once:
  pinMode(outputpin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(outputpin, HIGH);
  delay(1000);
  digitalWrite(outputpin, LOW);
  delay(1000);
}
```

Figure 7: Code modified to use an external LED on a different pin

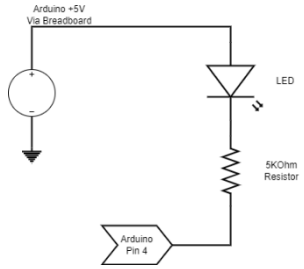

Figure 8: Digital output circuit. Now the pin is acting as current sink.

Reading a Digital Pin

1. Now we want to read the status of the digital pin using the digitalRead() function. We will use the pushbutton as our external input. If you look at the bottom of the pushbutton, you will see 4 leads. Two sets of the leads are always connected. When the button is pressed, all 4 leads are connected together. See Figure 9.
2. Now, remember we want to use a pull-up resistor to make sure the input is never floating. The schematic for this circuit is shown in Figure 10 and the breadboard is shown in Figure 12.
3. Now write a program that reads the status of pin 6 and prints out a message to the serial port depending on whether it his high or low.

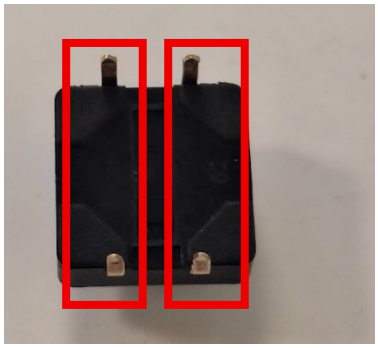

Figure 9: Pushbutton as seen from the bottom. The pins in the red boxes are always connected. When the button us pushed, all 4 pins are connected together.

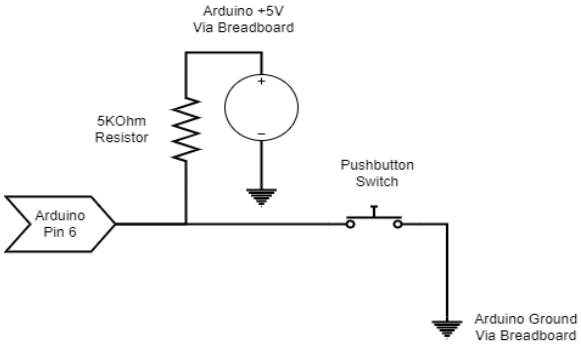

Figure 10: Input circuit using a pull-up resistor. Normally, the input is connected to +5V via the resistor, so the pin reads High. When the button is pressed, the input connects to ground directly, causing the pin to read low.

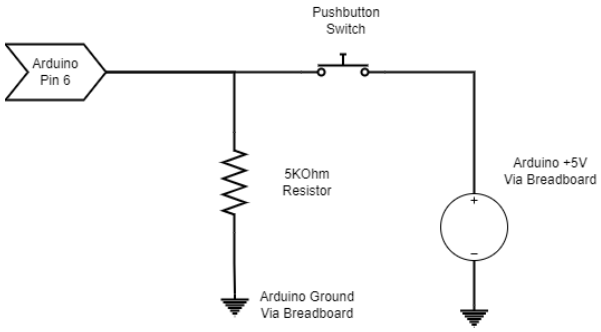

Figure 11: Input circuit using a pull-down resistor. In this case, the input will be low by default and change to high when the button is pressed.

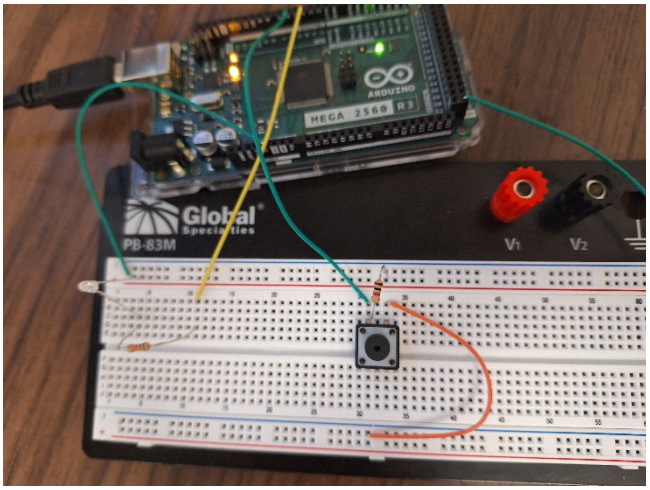

Figure 12: The breadboarded circuit using the pull up resistor. The LED from the previous section is still connected here can be ignored.

Don't forget we will need to set the baud rate for the serial port in the setup, and we will want a variable to store the result of our **digitalRead().** So the program should function as shown in Figure 13. The sample code is shown in Figure 14.

4. Once your code is uploaded, On printed repeatedly since that is the default input with the pullup resistor. Once the button is pressed the input should change to off.
5. Now modify your circuit to use a pull-down resistor as shown in Figure 11. Again, your previous code should work without modification. The behavior should reverse with "Off" being displayed by default and it should change to "On" when the button is pressed.

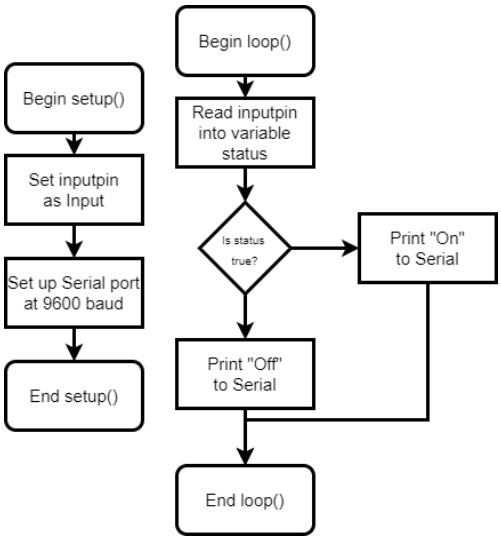

Figure 13: Flowchart for our digital read program. Notice we do not want any delays, digital read cannot remember the status, it only checks the status at the moment the function is called.

```
int inputpin=6;

void setup() {
  // put your setup code here, to run once:
  pinMode(inputpin, INPUT);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  int status;

  status=digitalRead(inputpin);
  if (status){
    Serial.println("On");
  }
  else{
    Serial.println("Off");
  }
}
```

Figure 14: Code to read the input on Pin 6 and then print an output message to the serial monitor.

Combining Digital Inputs and Outputs

1. For the final part of this activity, combine the input and output functions. You should construct the circuit shown in Figure 15 using the provided slide or toggle switch. The middle lead of the switch connects to the other sides, depending on the switch's position. That means you should connect the middle lead to the output pin and the two outer leads to +5V and GND.
2. The LED portion of the circuit is the same as the current source output circuit built previously.
3. Once your breadboard circuit is complete, your program should read the position of the switch and turn the LED on or off depending on that position, as shown in Figure 16.

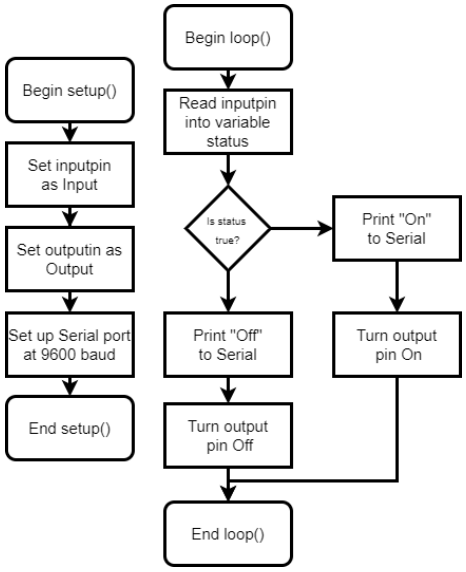

Figure 16: Flow chart for the combined input and output program.

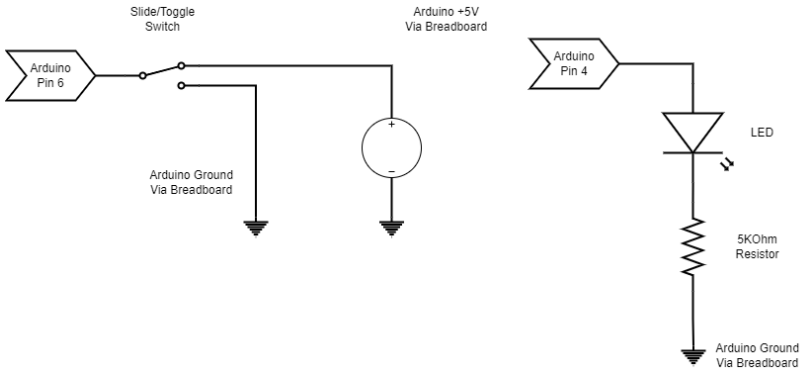

Figure 15: Combined input and output circuit. Because there is an active connection to +5V or GND in either switch position, pull-up/down resistors are not required.

References

[1] Arduino, "Arduino Mega Schematic," [Online]. Available: https://content.arduino.cc/assets/MEGA2560_Rev3e_sch.pdf.