Activity 4.2.2

Introduction to Microcontrollers

INTRODUCTION

Until now, your selection of input devices and output devices has been limited to the sensors and human input devices available in your classroom. In today's world of electronics, there are a tremendous number of other devices you could use in your designs.

In this activity you will create your first **programs** (sketches) to control systems with unique sensors, human input controls, motors, and **servos** that you may not have used previously. The ATmega328 **microcontroller** found on the Arduino™ UNO Microcontroller Board will be used to explore these controls and inputs.

Programming languages have their own grammar called "syntax". Programs written with the Ardiuno software are called Sketches. A **Sketch** (program written with Arduino) will contain: a title, **constants**, **variables**, setup() functions, and loop() functions.

If the syntax of a language is not followed, the program will not compile correctly. This means that no executable code will be produced. Fortunately, the Arduino integrated development environment (IDE) will provide error messages that will help you fix your "bad grammar"... called "syntax errors". One of the most common syntax errors that students make is forgetting that lines of code need to end with a semicolon.

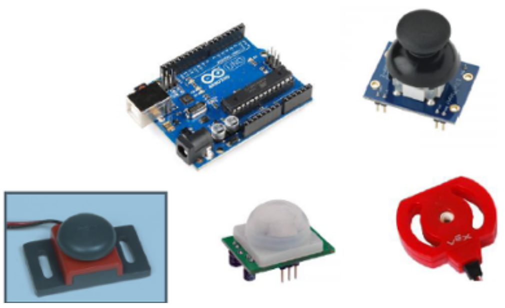
Structure of a Sketch (Program)

The *"Blink"* program used to test your Arduino is shown below to illustrate the parts of a program (sketch).

```
/*
Title: Description and comments.
Blink: This example make the LED at pin 13 blink.
This example code is in the public domain.
 */
// Constants: Constants won't change. They're used here to set the pin
const int ledPin = 13;          // constant ledPin assigned to pin 13
// Variables: Variables will change. They're used do assign variable n
                               // there are no variables in this exampl
// Setup: The setup routine runs once when you start or press reset:
void setup() {                  // put your setup code here, to run on
   pinMode(ledPin, OUTPUT);     // initialize the LED pin as an output
}
// Loop: The loop routine runs over and over again forever:
void loop() {                   // put your main code here, to run rep
  digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltag
  delay(1000);                  // wait for one second
  digitalWrite(ledPin, LOW);    // turn the LED off by making the volt
  delay(1000);                  // wait for one second
}
```

EQUIPMENT

- Parallax® student DE bundle with Arduino™
    - Arduino™ UNO Microcontroller Board
    - PIR Sensor (Passive Infra-Red)
    - Parallax® 2-Axis Joystick



- Arduino™ IDE Software
- Breadboard
- #22-gauge solid wire
- Resistors: 220 Ω; 10 kΩ
- LED
- Push-button or VEX bumper switch
- VEX potentiometer

Procedure

Sketch 1: *"Blink"*

1. Open the Arduino software and create the "*Blink*" sketch.

2. Title the project and add a description. Save.

3. Define **constant integer**, named "ledPin", assigned to pin **13**.

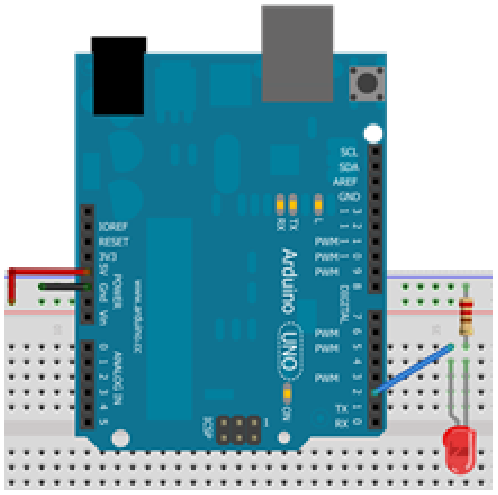
4. **void setup**()
   - Add a **pinMode** function so that variable **ledPin** is an OUTPUT.

5. **void loop**()
   - Add a **digitalWrite** function to make **ledPin** HIGH for **delay** 1000 ms.
   - Add a **digitalWrite** function to make **ledPin** LOW for **delay** 1000 ms.

6. Save the code as "Blink". Have your instructor verify that the circuit works as expected. The LED on the Arduino (pin 13) should be blinking.

Sketch 2: *"AltBlink"*

7. Create the circuit ***"AltBlink"*** using:



AltBlink Circuit

- An Arduino Microcontroller Board
- An LED
- 220 Ω resistor

8 Modify the “***Blink***” code to create a new sketch called “***AltBlink***”.

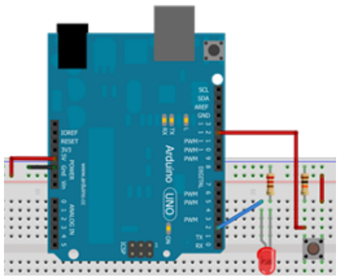
9 With LED A defined as the resistor on the Arduino, and LED B defined as the LED on the breadboard, make the two LEDs blink in an alternating pattern.

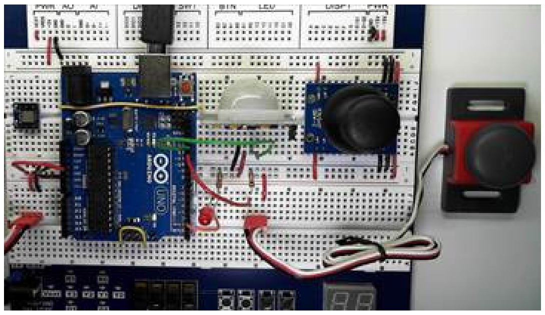
- LED A (on Arduino) is “off” when LED B (on breadboard) is “on”.
- LED A (on Arduino) is “on” when LED B (on breadboard) is “off”.

Sketch 3: “*Pushbutton*”

This sketch reads a VEX bumper switch (or push-button) and turns on an LED.

> **Note**: If a VEX bumper switch is not available, you may use a push-button on your Digital MiniSystem or a simple push-button on a breadboard as shown below.


Circuit 3 with pushbutton


Circuit 3 with VEX bumper switch

10 Using the Arduino, an LED, a VEX bumper switch, and a 10 kΩ resistor, create the circuit.

11 Open the Arduino software and create the bare minimum code.

12 Title the project and add a description. Save.

13 Constants

- Define **constant integer**, named **buttonPin**, assigned to pin **12**.
- Define **constant integer**, named **ledPin**, assigned to pin **2**.

14 Define variable **int buttonState** = **0;**

15 **void setup**()

- Add a **pinMode** function so that variable **ledPin** is an OUTPUT.
- Add a **pinMode** function so that variable **buttonPin** is an INPUT.

16 **void loop**()

- Read the input pin **buttonState** = **digitalRead** (**buttonPin**);
    - If (buttonState == HIGH), turn on the LED.
    - If (buttonState == LOW), turn off the LED.

17 Have your instructor verify that the circuit works as expected.

Sketch 4: “*DigitalReadSerial*”

This sketch reads a push-button and prints the outputs to the serial monitor. The wiring for this circuit is the same, and this new sketch can be created by modifying **Sketch 2** “***Pushbutton***”.

18 Open the Arduino software and create the bare minimum code.

19 Title the project and add a description. Save.

20 Define **constant integer**, named **buttonPin**, assigned to pin 1**2**.

21 Define variable **int buttonState** = **0;**

22 **void setup**()

- Add **Serial.begin (9600);** to initialize serial communication at 9600 b/s.
- Add **pinMode** function so that variable **buttonPin** is an INPUT.

23 **void loop**()

a. Read the input pin.
   buttonState = **digitalRead** (**buttonPin**);
b. Print the state of the button to the monitor.
   Serial.println (**buttonState**);
c. Add **delay**(1)**;** so you can see the change when it is writing to the monitor.

24 Open the serial monitor (Select Tools then Serial Monitor) then have your instructor verify that the circuit works as expected.

Sketch 5: “*DigitalReadSerialLED*”

25 From what you have learned create a new program **“*DigitalReadSerialLED*”** that that prints the digital output of a pushbutton to the Arduino serial monitor and has an LED indicator showing when the signal is high.

26 Have your instructor verify that the circuit works as expected.

Sketch 5: *"PIRDigitalReadSerial"*

Input Sensor: PIR Sensor

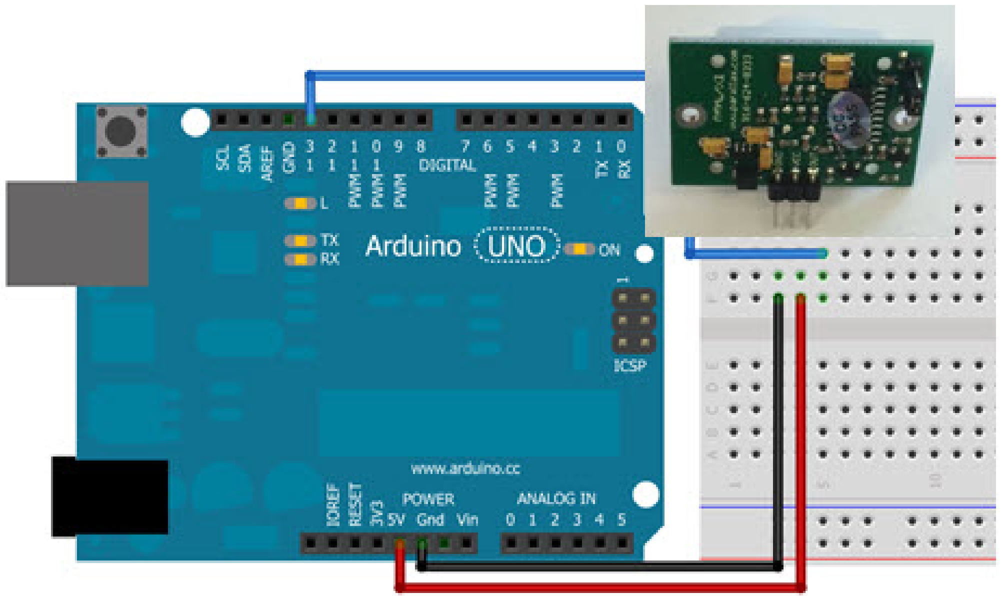
The Passive Infra-Red (PIR) Sensor is a pyroelectric device that detects motion by measuring changes in the infrared (heat) levels emitted by surrounding objects. This motion can be detected by checking for a sudden change in the surrounding IR patterns. When motion is detected, the PIR sensor outputs a high signal on its output pin. This logic signal can be read by a microcontroller or used to drive a transistor to switch a higher current load.

The PIR sensor has the following features:

- Detection range up to 20 feet away
- Single bit output
- Jumper selects single or continuous trigger output mode

Wire the PIR Sensor and Test

27 Wire in the following configuration. If motion is detected, the PIR dome lights up, sending a signal out to the LED wired to pin 13 on the Arduino. To test, make sure the sensor is in a location with no movement.

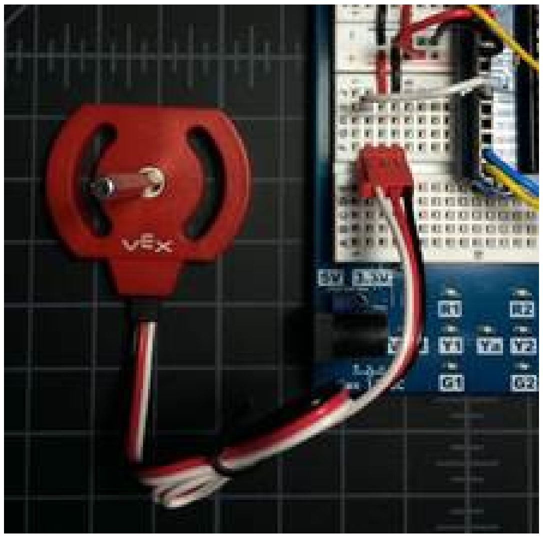


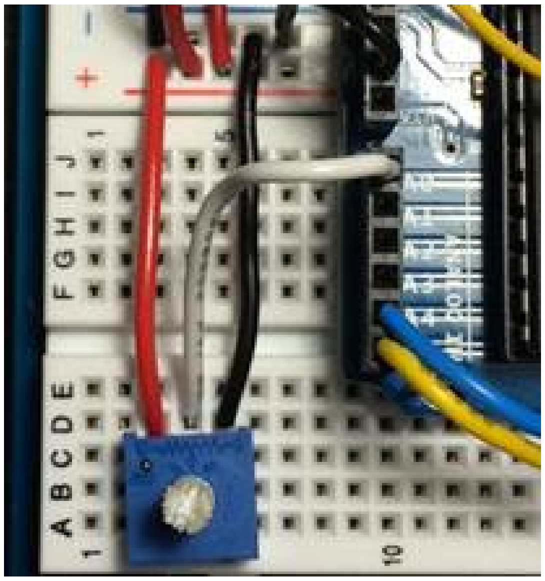
28 Using what you have learned, create a program (sketch) that will detect motion and print the state of the sensor to the Arduino Serial Monitor.

29 Have your instructor verify that the circuit works as expected.

Sketch 6: *"AnalogReadSerial"*

A potentiometer is a simple mechanical device that provides a varying amount of resistance when its shaft is turned. By passing voltage through a potentiometer and into an analog input on your Arduino, it is possible to measure the amount of resistance produced by a potentiometer (referred to as "pot" in notation) as an analog value. In this example, you will monitor the state of your potentiometer after establishing serial communication between your Arduino and your computer. This example is public domain; for more support, you can find this example among those on the Arduino website.


VEX Potentiometer with PWM


Potentiometer

30 Using the Arduino and a VEX and potentiometer, create the circuit.

31 Open the Arduino software and create the bare minimum code.

32 Title the project and add a description. Save.

33 Define **constant integer**, named **potPin**, assigned to pin A0.

34 Define variable **int potValue** = **analogRead (potPin);**

35 **void setup**()

- Add **Serial.begin** at 9600 b/s to initialize serial communication.

36 **void loop**()

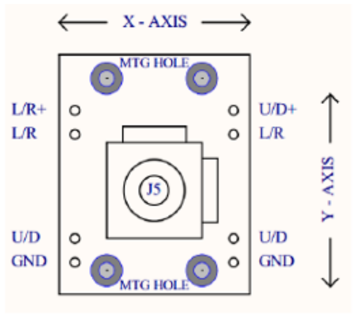
- Read the input on the analog pin.
  potValue = **analogRead (A0);**
- Print the state of the potentiometer to the monitor.
  Serial.println (potValue);

- Add **delay**(1)**;** so you can see the change when it is writing to the monitor.

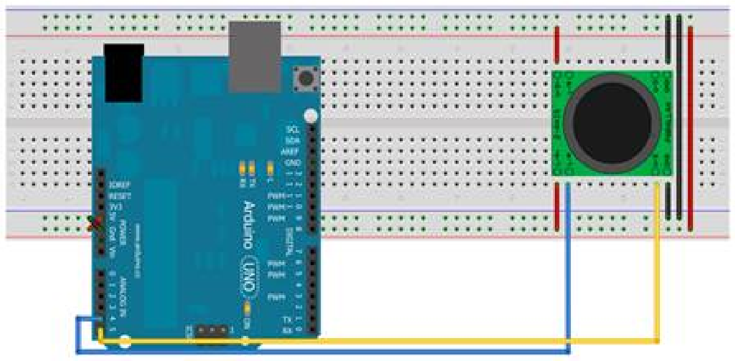
37 Have your instructor verify that the circuit works as expected.

Sketch 7: *"Joystick"*

The 2-Axis Joystick contains two independent potentiometers (one per axis) for reporting the joystick's position, with wiring options for voltage or resistance outputs.



2-Axis Joystick	Microcontroller Board
L/R+	5V
U/D+	5V
GND	GND
U/D U/D	A0
L/R L/R	A1



38 Based on what you learned in **Sketch 6 *"AnalogReadSerial"***, create a new sketch that will read the analog value for each axis and print those values to the Arduino serial monitor.

39 To distinguish which value you are reading, you will want to add a:

- **Serial.print("Up/Down value = ");**
  // words in quotes print on the serial monitor
- **Serial.print("Left/Right value = ");**
  // words in quotes print on the serial monitor

40 When the joystick is sitting in the middle, what are the values roughly?

41 When you move the joystick, what are roughly the maximum value and the minimum value you see?

42 What do you think this range should be exactly? (Hint: Think base 2 numbers.)

43 Have your instructor verify that the circuit works as expected.

Libraries and Communities

It is good practice to keep all your programs (sketches) organized so that you can use/build upon them in the future. Make sure you have captured the seven introductory programs and turned them in to your instructor in the format your instructor requested.

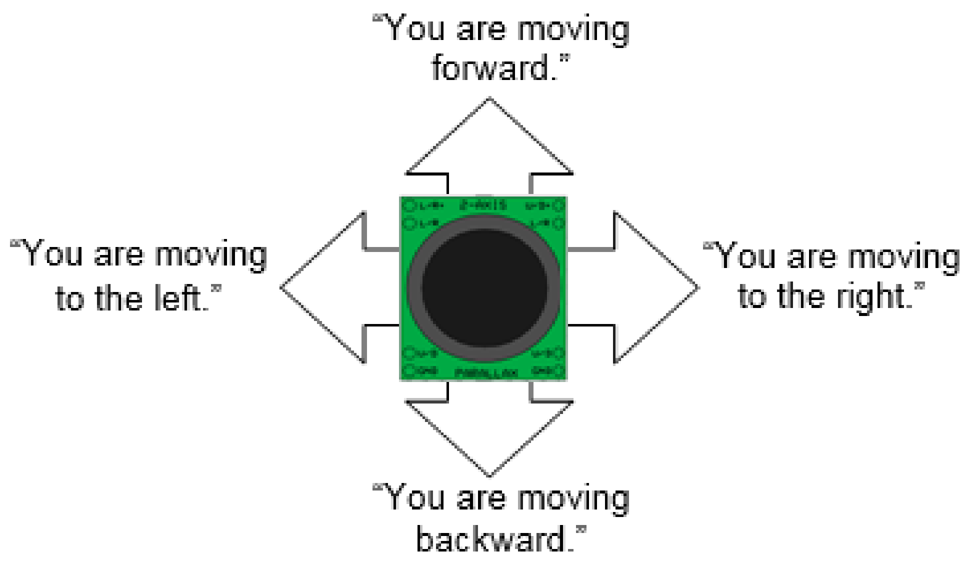
Because Arduino is an open-source community, there are hundreds of examples of code online you can use for your own projects.

CONCLUSION

1 Looking around the room or building you are in, identify 3–5 devices that most likely have a microcontroller embedded in them.

2 What are some of the major parts to a program or sketch?

3 Without worrying about syntax, what conditional statement would you write to create the following outputs on the serial monitor based on the inputs of the 2-Axis Joystick.


"You are moving forward."
"You are moving to the left."
"You are moving to the right."
"You are moving backward."
2-AXIS
PARALLAX

Proceed to next activity