DA274A: Internet of Things and People

Assignment 1

Introduction to Arduino
(Proximity Sensing and Vibration Motor)

Majid Ashouri
November 2018

What is Arduino?

The Arduino (shown in Figure 1) is a relatively inexpensive, yet versatile open-source microcontroller. It is designed to facilitate interaction with the physical world via sensors while being able to perform calculations and various functions. The Arduino can be connected to a computer via a USB cable and programmed using a simplified version of the C programming language, and it has both analog and digital pins from or to which it can read or write values. The maximum voltage that it is able to supply is 5V; thus, a "HIGH" digital pin corresponds to 5V, while a "LOW" digital pin corresponds to 0V. There are many "shields" and sensors that are designed for interaction with the Arduino, or microcontrollers in general. The Arduino can read values from sensors or other inputs, and it can also write values to other components based on computations given by the program.

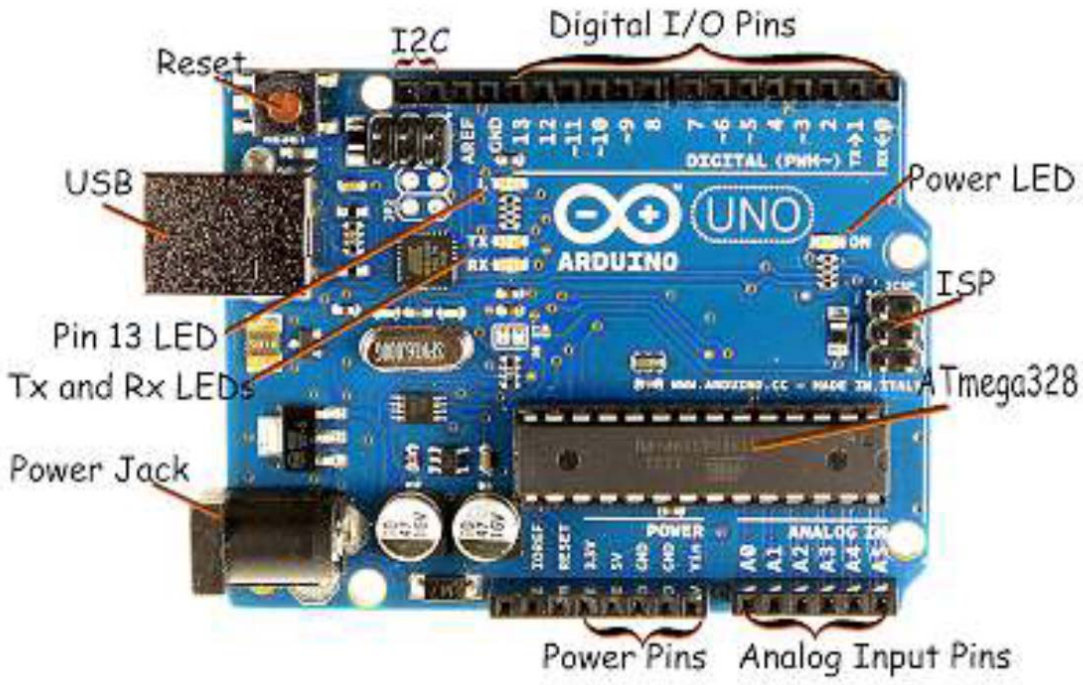

Figure 1. Arduino Uno

Arduino is one of the most popular microcontrollers on the market. Its ease of use, extensive software library and most importantly, its low cost have come to make it as popular as it is today. Many projects using the Arduino can be found on http://www.hackaday.com/. In order to start having fun with the Arduino, free software can be found at: http://arduino.cc/en/Main/Software for

Macs, Windows and Linux operating systems. This website also provides tons of easy tutorials for you to start. Tutorials can be found at: http://arduino.cc/en/Tutorial/HomePage.

Goals

- Learn how to program the Arduino Uno and use basic functions such as delay, pinMode, digitalWrite, digitalRead, analogRead, etc.
- Digital/analog sensing on the Arduino Uno
- Proximity sensing

Assignments

Task 1: Flashing an LED connected to the pin D1

In this section, you'll learn how to use Arduino Uno to flash an LED connected to digital pin 13.

1- Connect a resistor and an LED to the pin 13 and GND pin of the Arduino Uno using a breadboard:

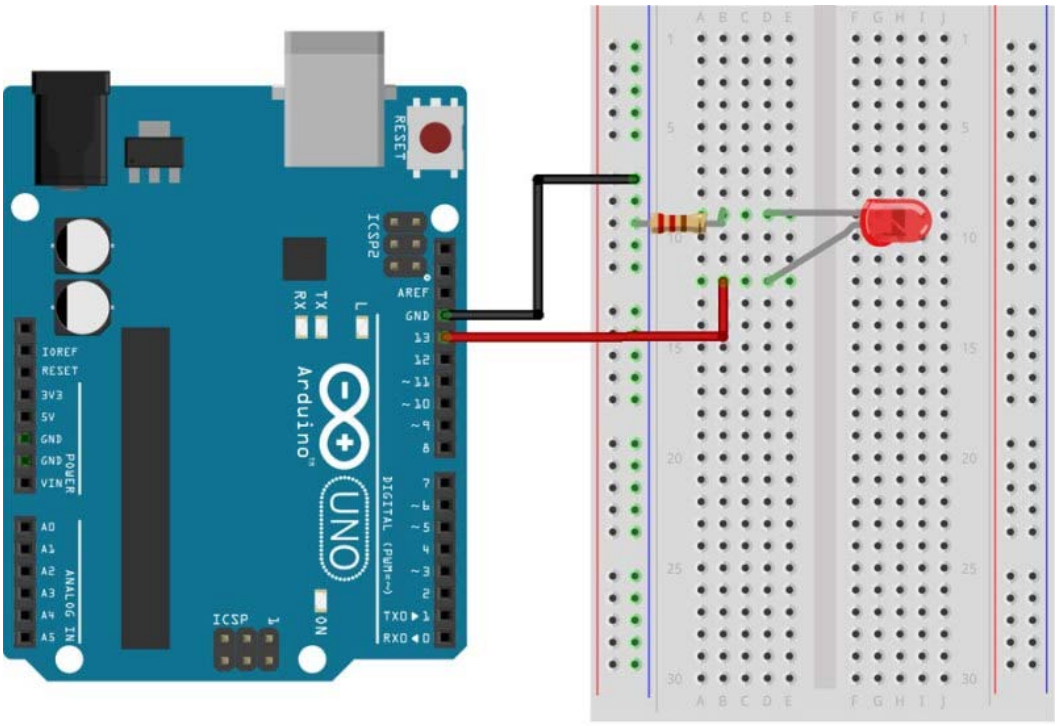

Figure 3. Connect an LED to Arduino Uno

2- Go to Start Programs and launch the Arduino IDE

3- Copy/Write the below code in the IDE window.

```
void setup() {
// initialize the digital pin as an output.
// Define Pin D1 as output:
pinMode(13, OUTPUT);
}

void loop() {
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(13, LOW); // set the LED off
delay(1000); // wait for a second
}
```

Notice that text following two slashes // or between /* */ is grayed-out and is not read by the Arduino; these are known as comments and serve to help the reader understand the code.

4- Go to the Tools menu of the Arduino IDE, and select the Serial Port (COM Port) to which the Arduino Board is connected to. This should be the last one (not COM1 or COM3). Make sure that the Arduino is plugged into one of the USB ports; otherwise you will only see COM1 and COM3.

5- Go to Tools > Board. Make sure the Arduino Uno 0.9 (ESP-12 Module) is selected.

6- Use the Upload icon of the Arduino IDE to compile and upload the program to the Board.

7- If your upload was successful, you will get the "… [%100] " message.

8- Look at your LED. The LED should start flashing on and off at a frequency of 0.5 Hz. In the setup function, the pinMode function is used to specify that pin D1 is being used as an output. Within the loop function of this code, pin 13 is written HIGH (pin 13 becomes a 5-V source) using the digitalWrite function. Next, the program waits for one second using the delay() function, which takes an argument in milliseconds. Then, pin 13 is written LOW (0V). The program again waits (does nothing) for a second. The sequence of events within loop will be repeated forever as long as the Arduino Uno is supplied with power, while the events in setup occur only once when the program starts running. Now you can modify the code so that the flashing rate is 1 Hz (1 cycle per second). The breadboard that you used to connect the LED to the Arduino Uno gives you a "workspace" on top of the microcontroller to build circuits that you can interface with the digital and analog pins.

Task 2: Testing Vibrator Motor

For this task you should work with vibrator motor instead of LED and try to vibrate it. You can follow the same procedure and the code you used in task 1, but you just need to remove resistor from the board. You can also change the delay.

Task 3: Proximity sensing using a PING range finder sensor

The PING range finder is an ultrasound sensor that is able to detect objects from 3 cm up to 400 cm distance. The sensor counts with 4 pins, two are dedicated to power and ground, while the other two used as input (Echo) and output (Trig). The pin dedicated to make the readings has to be shifting configuration from input to output according to the PING specification sheet. First we have to send a pulse that will make the sensor send an ultrasound tone and wait for an echo. Once the tone is received back, the sensor will send a pulse over the same pin as earlier. The width of that pulse will determine the distance to the object.

1- Use the bredboard to connect the sensor to the Arduino Uno according to the below circuit (Figure4)

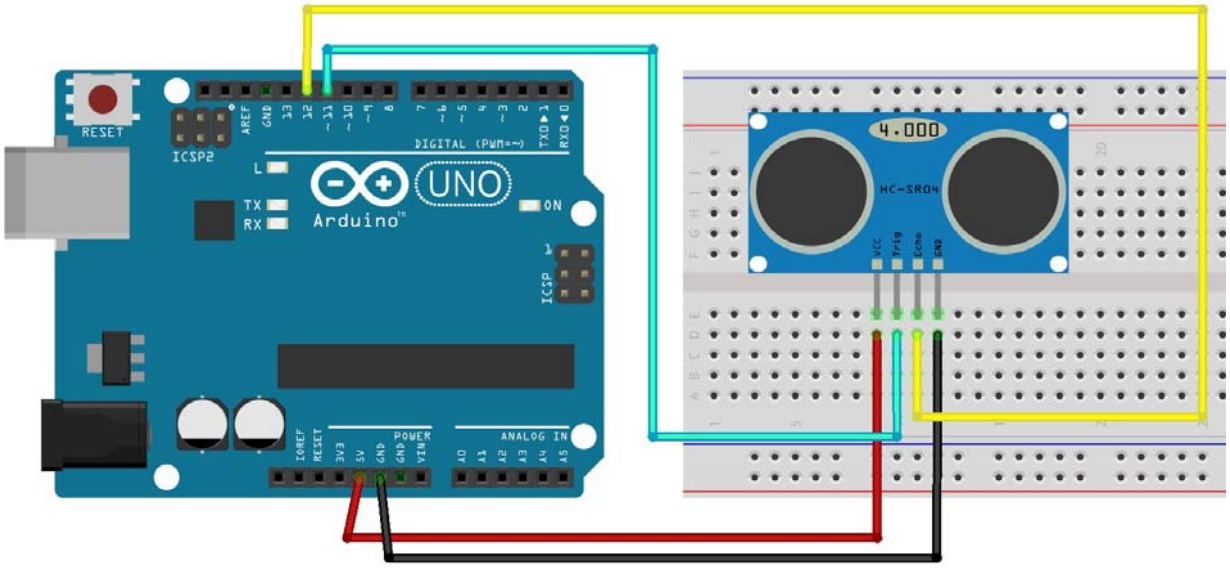

Figure 4. Connecting the PING sensor to the Arduino Uno

2- Copy and paste the below code to the Arduino IDE and upload the code.

```
/** from: https://codebender.cc/sketch:356078#HC-SR04%20Ultrasonic%20Sensor%20Example.ino
* HC-SR04 Demo
* Demonstration of the HC-SR04 Ultrasonic Sensor
```

```
* Date: August 3, 2016
*
* Description:
* Connect the ultrasonic sensor to the Arduino Uno as per the
* hardware connections below. Run the sketch and open a serial
* monitor. The distance read from the sensor will be displayed
* in centimeters and inches.
*
* Hardware Connections:
* Arduino | HC-SR04
* ------------------
* 5V | VCC
* 11 | Trig
* 12 | Echo
* GND | GND
*
* License:
* Public Domain
*/
// Pins
const int TRIG_PIN = 11;
const int ECHO_PIN = 12;
// Anything over 400 cm (23200 us pulse) is "out of range"
const unsigned int MAX_DIST = 23200;
void setup() {
// The Trigger pin will tell the sensor to range find, since the default mode is input we don’t need to define
Echo pin as
input
pinMode(TRIG_PIN, OUTPUT);
digitalWrite(TRIG_PIN, LOW);
// We'll use the serial monitor to view the sensor output
Serial.begin(9600);
}
void loop() {

unsigned long t1;
unsigned long t2;
unsigned long pulse_width;
float cm;
float inches;
// Hold the trigger pin high for at least 10 us
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
// Wait for pulse on echo pin
while ( digitalRead(ECHO_PIN) == 0 );
// Measure how long the echo pin was held high (pulse width)
// Note: the micros() counter will overflow after ~70 min
t1 = micros();
while ( digitalRead(ECHO_PIN) == 1);
t2 = micros();
pulse_width = t2 - t1;
// Calculate distance in centimeters and inches. The constants
// are found in the datasheet, and calculated from the assumed speed
```

```
//of sound in air at sea level (~340 m/s).
cm = pulse_width / 58.0;
inches = pulse_width / 148.0;
// Print out results
if ( pulse_width > MAX_DIST ) {
Serial.println("Out of range");
} else {
Serial.print(cm);
Serial.print(" cm \t");
Serial.print(inches);
Serial.println(" in");
}
// Wait at least 60ms before next measurement
delay(60);
}
```

3- Open the serial port in the menu and set the baudrate on 9600. Now you should be able to see the output of the sensor in serial port.

Mandatory Demo

In order to pass the lab 1 you need to demo the result of the following task to the lab assistant:

Define a minimum proximity in such a way that if something gets closer than that minimum distance to the ping proximity sensor the LED lights up, and vibrator starts vibrating.