Microcontrollers Lab - 10636496

Arduino Experiment #1: An Introduction

Prepared by: DR. MANAR QAMHIEH

Objectives:

This lab will focus on making a rapid prototype and learning the Arduino environment, and to gain a basic familiarity with the Arduino environment. Arduinos are a standard Atmel microcontroller with a really simplified programming environment, and they are really targeted toward non-programmers.

This environment is important to learn for two reasons:

- It is a powerful tool for quickly prototyping devices and getting basic interfacing working.
- It provides some really solid interfaces so that people who don't understand the hardware can use it.

In this lab, we will be using Arduino Mega 2560 in addition to many peripherals such as sensors, controlling drivers and communication protocol.

Resources and Documentation:

- Arduino IDE: https://www.arduino.cc/en/software (download) https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/ (tutorial)
- Arduino Documentation: https://docs.arduino.cc/
- Arduino Mege 2560: https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf

Getting Started with Arduino IDE

The Arduino Integrated Development Environment (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 hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension **.ino**.

The Arduino program consists of two functions called *setup* and *loop*. The *setup()* function initializes and sets the initial values, while the *loop()* function loops consecutively, allowing your program to change and respond.

```
void setup() {
 // put your setup code here, to run once:
}
void loop() {
 // put your main code here, to run repeatedly:
}
```

Note that we won't need any includes or defines for this part; you only need to define a setup and loop function. This is because the Arduino IDE transforms your sketch into a valid C++ program by adding the following lines of code:

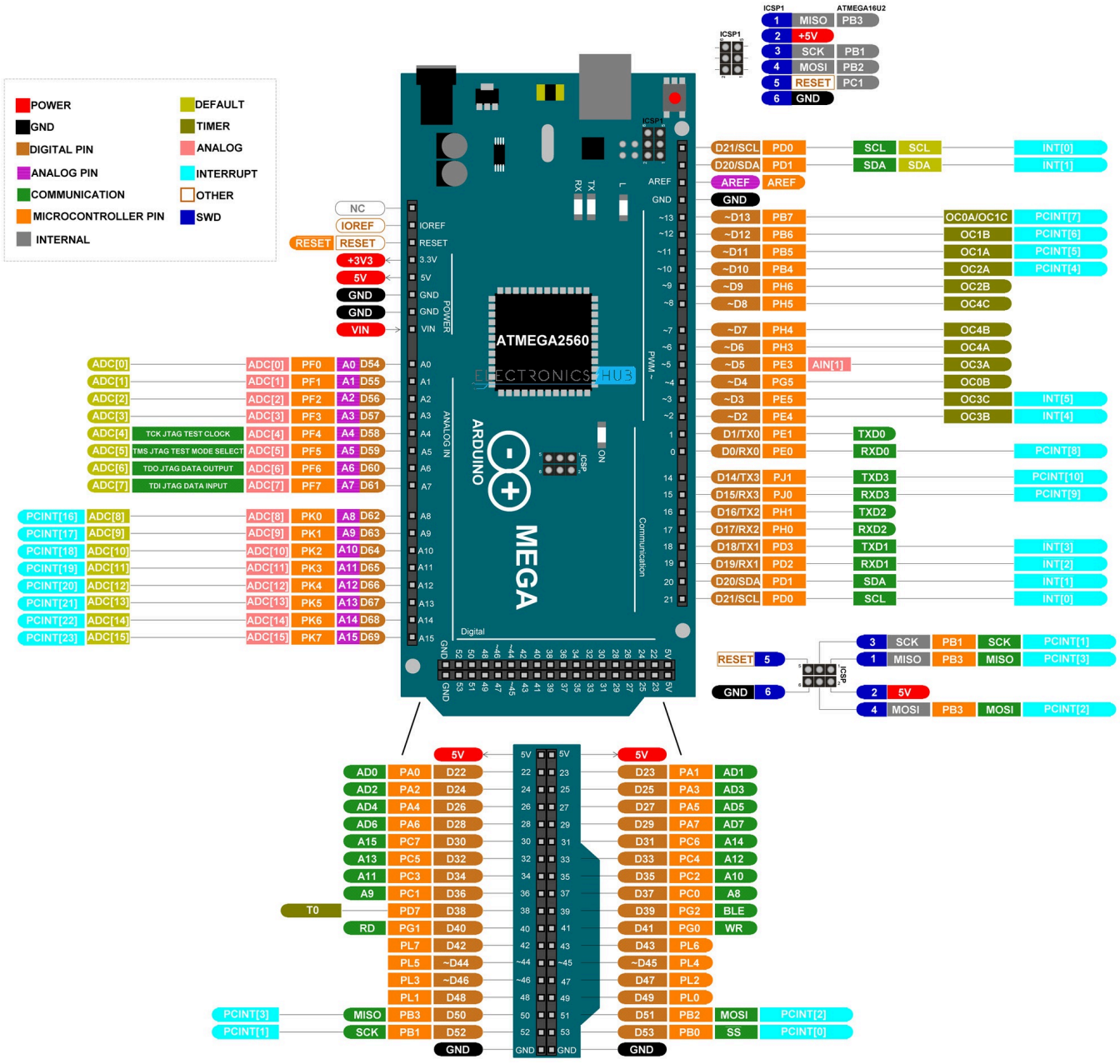
```
#include "WProgram.h"  // Definitions relevant to Arduino
void main ( ) {
      setup();
      while(1) {
            loop();
      }
}
```

- Once the code is written, you need to check to see if it compiles by pressing the icon named **“Verify”.** This will check for errors and compile your code.
- After the program compiles successfully, we still need to upload the program to the board.
- Connect the Arduino via the USB port. Select the correct **board** and **port** (might require you to go to the control panel and look at the device drivers).
- Then click the second icon on the **upload** icon . If you get an error, check to be sure you’ve selected the correct device and port. If you cannot select a different COM port and cannot program your board, you may need to quit Arduino and restart it with the board unplugged.

Introduction to Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

Primary processor of Arduino Mega 2560 board is ATmega2560 chip which operates at a frequency of 16 MHz. It accommodates a large number of input and output lines which gives the provision of interfacing many external devices. The board also features a USB serial processor ATmega16U2 which acts an interface between the USB input signals and the main processor.



The objective of this lab experiment is to get familiar with the Arduino environment by testing the Arduino Mega 2560, and also the programming language and get to know the available functions and libraries (https://www.arduino.cc/reference/en/). There are many Examples available from the Arduino IDE to help you get started.

Task 1:

Write a simple code to blink the built-in LED in Arduino Mega (pin 13) infinitely. There is a constant called LED_BUILTIN that is specified in every board descriptor file.
You need to get familiar with the **pinMode**, **digitalWrite** and **delay** functions in order to write your code.

Task 2: Serial Monitor

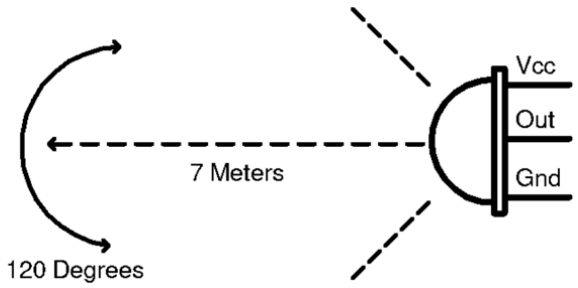
The Serial Monitor is an essential tool when creating projects with Arduino. It can be used as a debugging tool, testing out concepts or to communicate directly with the Arduino board. The Arduino IDE 2 has the Serial Monitor tool integrated with the editor, which means that no external window is opened when using the Serial Monitor.

In order to use the serial monitor, you need to include some configurations so that the MCU board can communicate with the computer. Mainly, we need to set a **baud rate**, which is done by writing Serial.begin(9600);. Here, the 9600 represents the baud rate, which is the maximum bits per seconds that can be transferred. Then, in order to print a message to the serial monitor, you can use the Serial.println function.

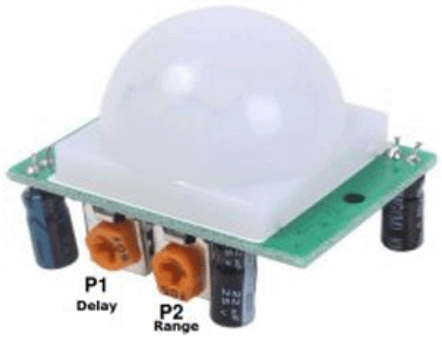
- Write a code to print “Hello World” on the serial monitr periodically every 1 second.

Task 3: PIR Sensor

A PIR sensor detects the change in infrared radiation (heat) caused by an object that generates or reflects heat as it moves within the sensor’s detection field. The PIR sensor’s detection field is characterized by detection distance and detection angle.



A typical PIR sensor has at least 3 connection pins: supply voltage pin (VCC), output pin (OUT), and circuit ground (GND). When an object enters the detection field, the PIR output transitions from it's untriggered (LOW) to it's triggered state (HIGH). Trimmer P1 controls the output delay time and trimmer P2 controls the detection range. To increase or decrease either setting, the appropriate trimmer is turned clockwise or counterclockwise respectively.



- Write a code to read the PIR sensor. Whenever a motion is detected, send a message to the serial monitor and turn on the built-in LED. You will need the function digitalRead in your code.

Task 4: Basic Servo Control

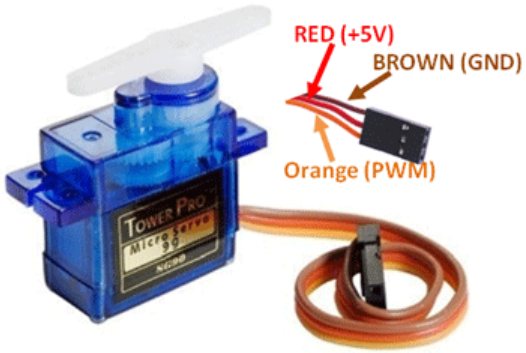
Standard servo motors are actuators that allow for precise control of position (angle). A typical characteristic is that the angle of the motor is 0 - 180 degrees (one half of a rotation).

A standard servo motor are essentially just a DC motor, but with some extra features:

- **Control circuit** for controlling the motor, e.g. setting the angle.
- **Gears** that transform speed into torque.
- **Potentiometer** that keeps track of its angle. This makes it possible for the servo to "know where it is".

Almost all servos come with a set of 3 wires. These are PWR-RED (5v), GND-BLACK (GND) and Signal-ORANGE (connects to a digital pin on the Arduino (typically 9).

In order to control the Servo Motor, you can use the Servo.h library, which includes functions like attach and write.

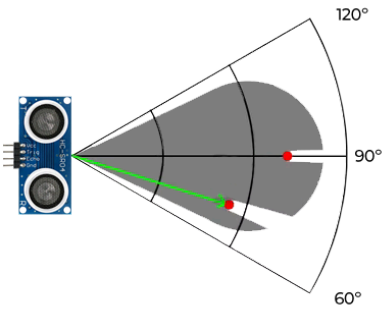


- Write a code to control the rotation of a Servo motor, and make it goes back and forth across 180 degrees, using a for loop.

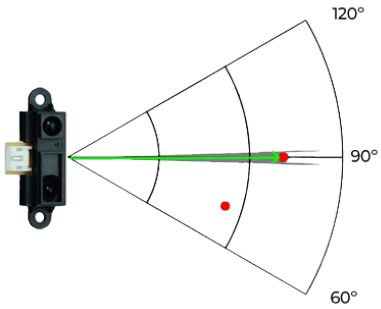
Task 5: Obstacle Detection: IR Sensor vs. Ultrasonic Sensor

The objective of this task is to compare the performance of Infrared (IR) sensors and Ultrasonic sensors in detecting and measuring the distance of an object.

The biggest difference between IR sensor vs. ultrasonic sensors is the way in which the sensor works. Ultrasonic sensors use sound waves (echolocation) to measure how far away you are from an object. On the other hand, IR sensors use Infrared light to determine whether or not an object is present.



Usually, ultrasonic sensors provide more reliable and accurate data than IR sensors, and it can provide a numerical representation of distance of the obstacle. However, an IR sensor is easier to implement in case of basic obstacle detection.



- **Testing the IR Sensor**

Similar to PIR sensor, The IR sensor has a 3-pin connector consisting of Power (3.3v/5v), GND and digital output (LOW indicates no motion is detected; HIGH means motion is detected).

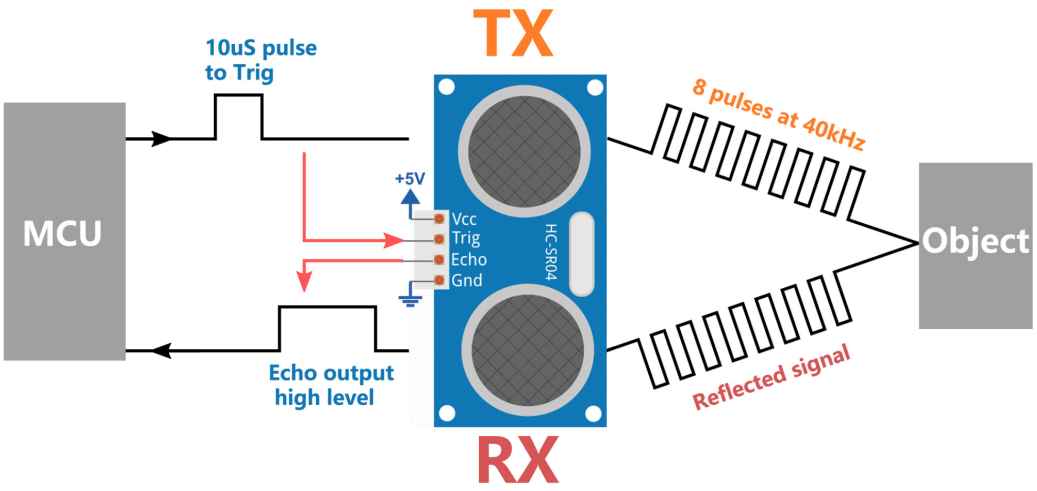
Note that the effective distance range of the IR sensor is 2 ~ 30cm. The sensor's detection range can be adjusted by adjusting the potentiometer.

Write a code to test the IR sensor and how it can be used to detect an obstacle.

- **Testing the Ultrasonic sensor**

The HC-SR04 is an affordable and easy to use distance measuring sensor which has a range from 2cm to 400cm. The sensor is composed of two ultrasonic transducers. One is transmitter which outputs ultrasonic sound pulses and the other is receiver which listens for reflected waves.

The sensor has 4 pins. VCC and GND go to 5V and GND pins on the Arduino, and the Trig and Echo go to any digital Arduino pin. Using the Trig pin we send the ultrasound wave from the transmitter, and with the Echo pin we listen for the reflected signal.



In order to generate the ultrasound, we need to set the Trig pin on a High State for 10 µs. That will send out an 8 cycle ultrasonic burst (40 000 Hz) which will travel at the speed of sound. The Echo pins goes high right away after that 8 cycle ultrasonic burst is sent, and it starts listening or waiting for that wave to be reflected from an object.

- If there is no object or reflected pulse, the Echo pin will time-out after 38ms and get back to low state.

- If we receive a reflected pulse, the Echo pin will go down sooner than those 38ms. According to the amount of time the Echo pin was HIGH, we can determine the distance the sound wave traveled, thus the distance from the sensor to the object.

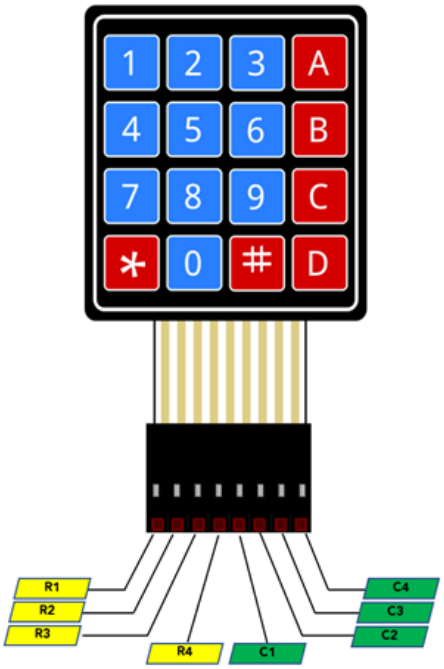
Based on the received Time (from the Echo pulse), we can calculate the distance by:

Distance = Speed x Time / 2

Where the Speed is 340 m/s (the speed of sound), and the ping travels out and back, so to find the distance of the object we take half of the distance travelled.

- Write a code to test the ultrasonic sensor, and display the distance on the Serial Monitor. You will need the functions delayMicroseconds and pulseIn in your code.

Task 6: Keypad



The pinout for an Arduino 4×4 keypad typically consists of 8 pins, 4 for the rows and 4 for the columns. These pins can go to any digital Arduino pin.

To read input from Keypad first we have to install the **Keypad Library** in Arduino IDE. After that, using the digital pins and library code, we can read data from the keypad.

After installing the library, refer to its examples (such as HelloKeypad.ino) for more information on how to use it, and write a code to print the pressed key on the serial monitor.