# Arduino Lab 1 –Compass to Servo

In this lab you will be learning how to use two sensors, the compass and the servo motor. The compass device can compute the bearing it faces, it outputs the direction as an angle from 0 to 360. For example 0 degrees would be North, and East would be 90 degrees. The servo motor can rotate objects connected to it, the Arduino can control it by specifying a degree in which to face.

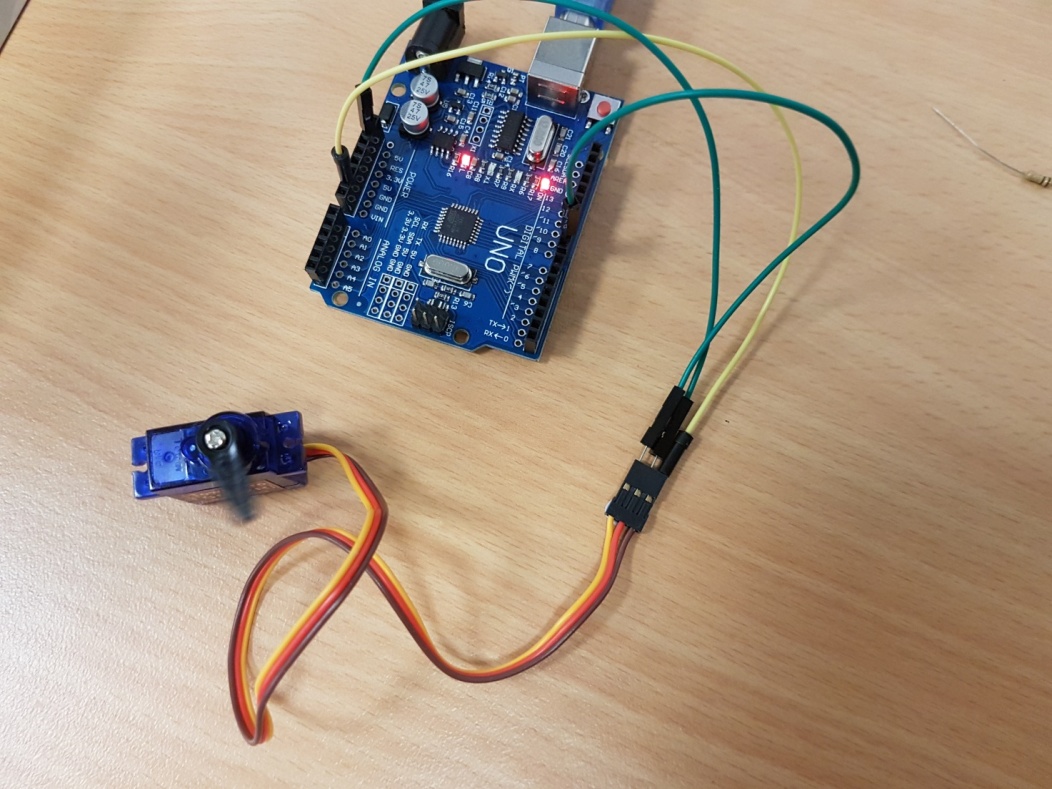
In this lab you will have to use the compass to control your servo motor.

Steps:

1. Do the Servo motor tutorial
2. Do the Compass tutorial
3. Attempt to do this lab, details are given below

## Servo Motor

The servo module requires the Servo.h library. Here we connect the Red pin to 5V and the Brown pin to GND. The yellow pin is a signal pin, we used digital pin 9 as the signal pin. The wiring is shown below.



First let’s take a look at our setup function, global variables and includes:

/\*

\* connect Red to 5v, Brown to gnd

\* connect Yellow to pin signalPin (9)

\*/

#include <Servo.h> //include Servo.h

void setup() {

Servo s; //create a servo object

int position = 45; //set the position to change to 45 degrees

s.attach(9); //attach to the signal pin (pin 9 in our case)

s.write(position); //set servo to point to an angle (variable position in this case)

}

void loop() {

}

This example is very simple. In this code, we are including Servo.h. This code provides us with a Servo object by which we can use some simple functions to work with the servo motor. In our setup function, we create a Servo variable named s. We also create an int variable position and set it to 45. Using our servo variable s, we attach our servo motor to digital pin 9 using s.attach(9). Finally we set the position for the servo to turn the s.write function, were we use the position variable to transmit this. The position variable may be any int value within the range 0 to 180. Play around with the variable position ad re-run your program to view the result.

Example 2.

int signalPin = 9;

/\*

\* connect Red to 5v, Brown to gnd

\* connect Yellow to pin signalPin (9)

\*/

#include <Servo.h>

Servo s; //create a servo object

int index;

int positions[6] = {0, 45, 90, 180, 50, 120}; //6 positions to go-to

void setup() {

s.attach(signalPin); //attach to the signal pin

index = 0;

}

void loop() {

s.write(positions[index]); //set servo to point to an angle in positions

delay(1000); //wait for 1 second

index = (index+1) % 6; //iterate index over the range [0,5] inclusive

}

This example is more complex, here we define positions as an array with 6 different angles. We setup our Servo object “s” outside the setup function so it may also be used in our loop function. We have another variable “index” which we will use to select the angle to use for in the array to give to the motor. In our loop function, at each stage of the loop we send the angle in positions at index “index” to the motor. Next we call the delay function and delay for 1000 milliseconds (1 second). Then we set index to be 1 value higher, but if index reaches 6 (outside the range 0 to 5 used for the array) then we set it back to 0. We do this with the index = (index + 1) % 6 line. If we printed the index variable at each stage of the loop it would look like this:

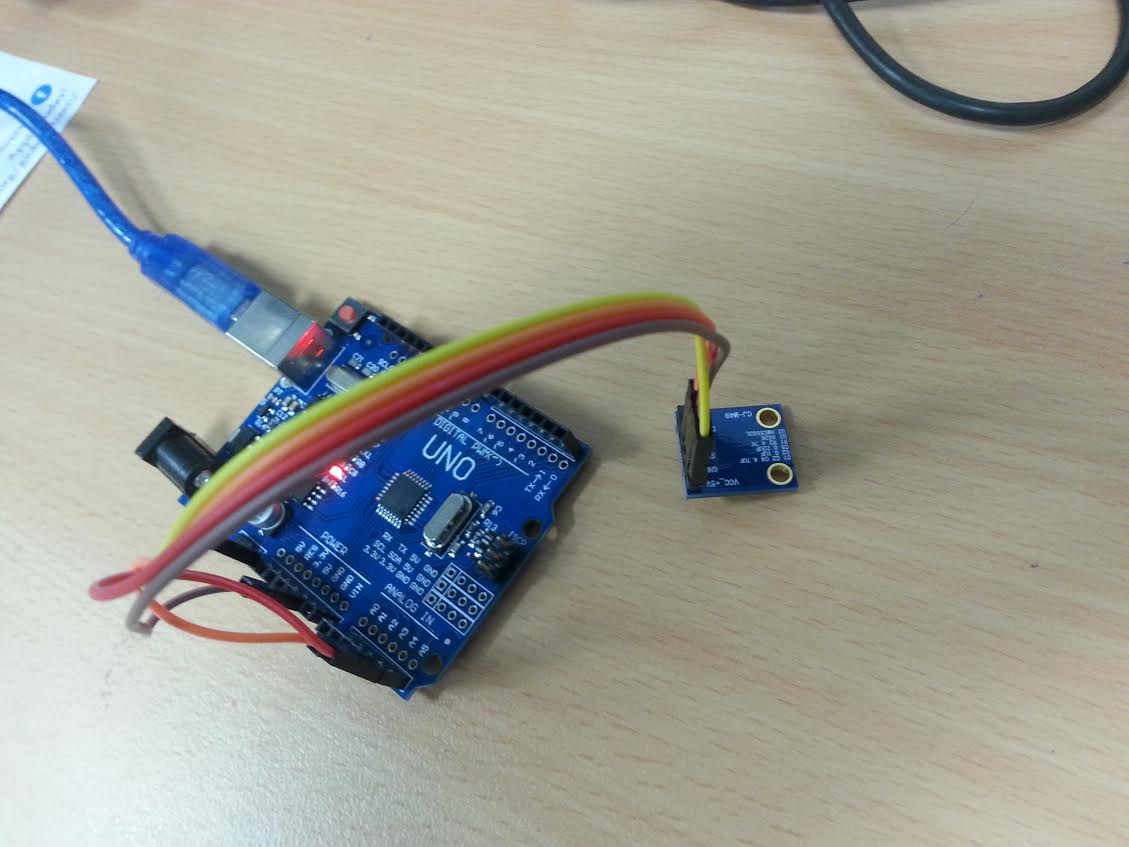
0 1 2 3 4 5 0 1 2 3 4 5 0 1 2 3 …

So therefore we would not select an angle in positions which does not exist.

Try changing the angles in positions, if you change the number of angles to N (where you choose N) remember to change the code saying “index = (index+1) % 6;” to “index = (index + 1) % N;” where N is the number you chose for the number of angles positions holds.

## Digital Compass Tutorial

The digital compass module requires a few libraries, namely: Wire, Adafruit\_Sensor and Adafruit\_Adafruit\_HMC5883\_U. You can install these libraries by downloading them and copying the folder into the c:/Arduino/libraries folder. I have put all the libraries you will need on my github at this url <https://github.com/lukes611/ArduinoTutes>. Simply click the dropdown for Clone or download and download the repository. Next, unzip it. Then copy the library folders you need (in the libs folder) to your C:/Arduino/libraries folder.

For connecting the compass I have connected the pin labelled 5V to one of the 5V pins on the Arduino and the GND pin to one of the GND pins on the Arduino. You must also connect the SCL pin to the Analogue 5 pin on the Arduino and the SDA pin to the Arduino Analogue 4 pin. The wiring is shown below.

First let’s take a look at our setup function, global variables and includes:

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_HMC5883\_U.h>

/\* Assign a unique ID to this sensor at the same time \*/

Adafruit\_HMC5883\_Unified mag = Adafruit\_HMC5883\_Unified(12345);

void setup(void)

{

Serial.begin(9600);

/\* Initialise the sensor \*/

if(!mag.begin())

{

Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");

while(1);

}

}

Of course we are including the necessary includes at the top. We will be using the Adafruit\_HMC5883\_Unified object from the Adafruit library in order to read the input from our compass. We name this variable mag, in our setup function we initialize our Serial object, and call begin() with our Adafruit object, it will return true if the initialization was correct. If it returns false we print an error message and go into an infinity while loop (never reaching our loop function).

Below is the accompanying loop function. In the setup function we obtain a new event by passing a pointer of a local event object to our mag object. This provides us with our x,y,z axis through variables: event.magnetic.x, event.magnetic.y and event.magnetic.z. Here we simply print them out to the serial monitor. If your compass is facing upwards (flat with the pins pointing towards the sky) then you can use the heading measurement. This is based on the x and y components of our vector, we simply find the angle which these two represent using the atan2 function, as in most languages this returns radians, so we therefore adjust for degrees for a human readable format.

void loop(void)

{

/\* Get a new sensor event \*/

sensors\_event\_t event;

mag.getEvent(&event);

/\* Display the results (magnetic vector values are in micro-Tesla (uT)) \*/

Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print(" ");

Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print(" ");

Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print(" ");Serial.println("uT");

// Hold the module so that Z is pointing 'up' and you can measure the heading with x&y

// Calculate heading when the magnetometer is level, then correct for signs of axis.

float heading = atan2(event.magnetic.y, event.magnetic.x);

// Convert radians to degrees for readability.

float headingDegrees = heading \* 180/M\_PI;

Serial.print("Heading (degrees): "); Serial.println(headingDegrees);

delay(500);

}

## Lab 1: Compass to Servo

If you haven't setup your development environment for the Arduino or run through the basics of using both the compass and the servo motor, please do so prior to beginning this lab. In this lab you will be required to re-direct the heading from the compass module into the direction of the servo motor.

Write a C-program which first extracts the compass heading in degrees (not radians). Next, send this angular value to the servo. This process is to repeat once every second. Use a global integer MapType which is either one or zero. If it is zero, whichever value (from 0 to 360) which the compass gives, output that value directly. If the MapType is one, then make sure the 0 to 360 value is mapped to 0 - 180. Use scaling (multiplication) to achieve this.

(Advanced) Next week you will study about functions. Functions allow the programmer to break up a program into modular pieces. Repeat this lab using functions, one function called get-heading has the signature:

int getHeading();

The other function set-Servo has the signature:

void setServo(int heading, int map\_type);

Your loop function may end up looking something like this:

void loop()

{

setServo(getHeading(), MapType), delay(1000);

}