ECE 1001/1002 Introduction to Robotics
Lab #7: Servo and Ultrasonic Distance

Objectives

Introduction to the servo motors, the ultrasonic distance sensor, and the serial display

Robots move, right? Even though that is not necessarily so, it's time to start making movement.

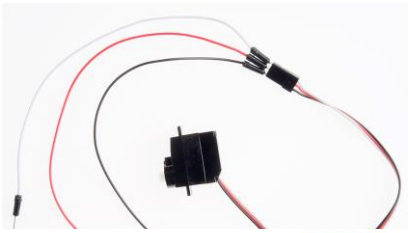
Your kit has two kinds of motors, a servo-motor which can move to well controlled positions suitable as an actuator and two dc motors suitable for drive wheels. This lab works with the servo motor.

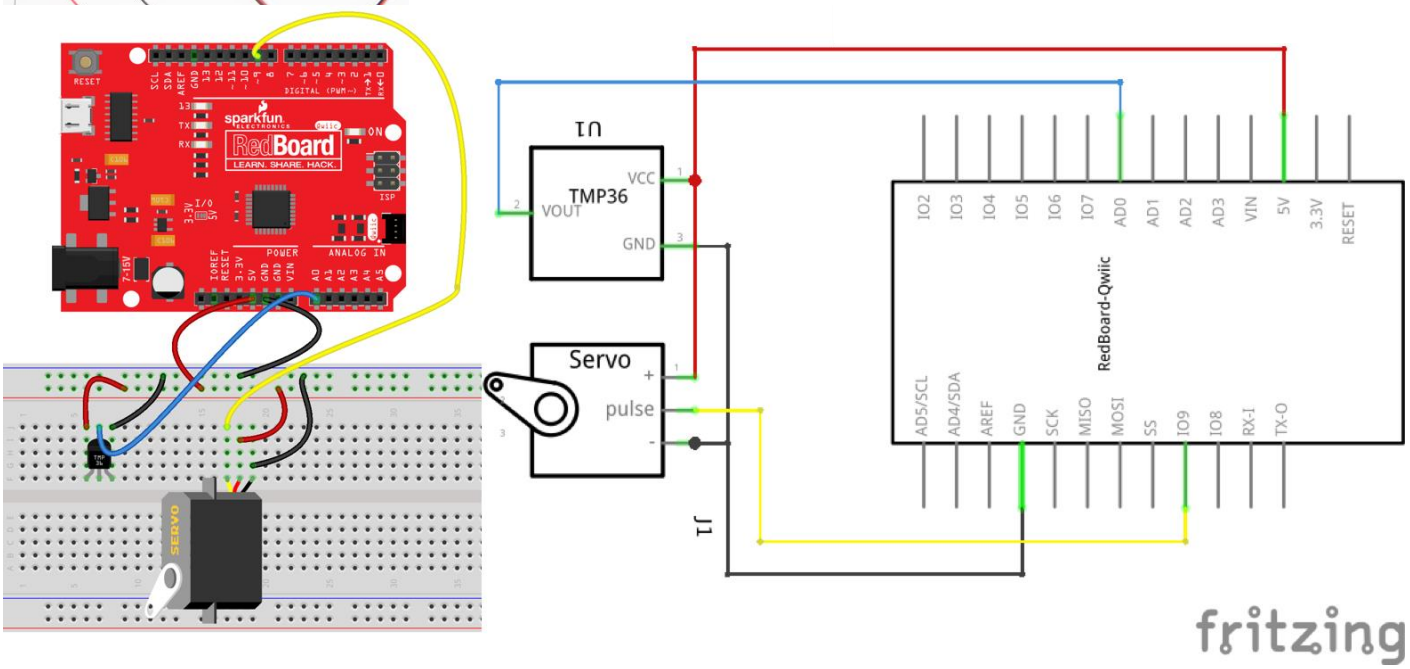
Servo motors are controlled with PWM signals, but the exact control method is complicated. It's been implemented for Arduino, and just as with the display, including a library (a "header file") enables you to write simple commands which will control a servo.

Requirements and Signoffs

Analog Temp Display

Build the Servo circuit as shown below, with the TMP36 temp sensor. You may want to cut a small piece of the double-sticky Velcro to hold your servo to a convent place (not on the breadboard, though), and install one of the mounts on the shaft of the Servo. The servo by itself looks like below. Note how jumper wires are inserted in the connector; pay special attention to the servo wire colors: red means 5V, black means Ground, and the third color means signal from the Arduino.





Analog Temp Display with Servo

Open the file, Lab7_temp_analog_servo_display.ino, and look at it. The program uses the servo library, so it must include <Servo.h>

The voltage and temperature float variables, measurements and calculations are re-used in this program. But the servo requires integer inputs. Mixing integer and floating point numbers can lead to unexpected errors. It's best to plan on a change or mix of variable types, C does this with a "type cast" which temporarily makes one variable type into another. The float TermperatureInC is converted temporarily into an integer variable and stored in the integer variable TemperaturePosition with this way:

TemperaturePosition= (int)TemperatureInC;

The (int) before TemperatureInC tells Arduino to just ignore the decimal and numbers after the decimal. That means if the temperature is 25.2 or 25.9 it will be converted to 25. But for this program, the temperature will move the Servo arm, and the loss of the precision won't be visible. It's important to explicitly plan and know when a conversion between variable types (like float to int) occurs. The program could have been re-written using all integer variable, but then it would mean I have two different programs with two different algorithms for converting voltage to temperature, and that having two different programs do (almost) the same thing can make even more work (or confusion) later on down the road when the program is used again.

The Servo moves the arm between 0 and 180. The temperature could vary over a very wide range, but for this we could say that the min temperature will be 0C and occur at 180 degrees (arrow pointing to the 6 o'clock position, while the maximum temp will be 50C with the arm pointing to the 12 o'clock position. This change is done with a map command with a syntax:

```
map(variable, fromLow, fromHigh, toLow, toHigh)
```

In this program, the value of ServoPosition is min 180 and max 0, while the TemperaturePostion is min 0 and max 50:

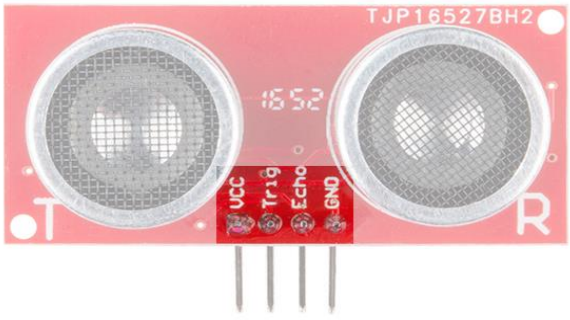
ServoPosition=map(TemperaturePosition, 0, 50, 180, 0);

Then the ServoPostion is written to the servo, and there is a 250 ms delay before the main loop repeats the measurement and position is adjusted.

Try it out! How quickly does it seem to respond? Is it smooth?

Ultrasonic Distance Measurement Hardware

Your kit includes an ultrasonic device for measuring distance. It works similar to how radar works, except it uses high frequency chirps we can't hear. It emits a chirp from the emitter on one side, and times how long it takes for an echo to return. With that information, it can calculate the distance to an object. There are plenty of issues, but the system works well-enough. The sensor looks like this:



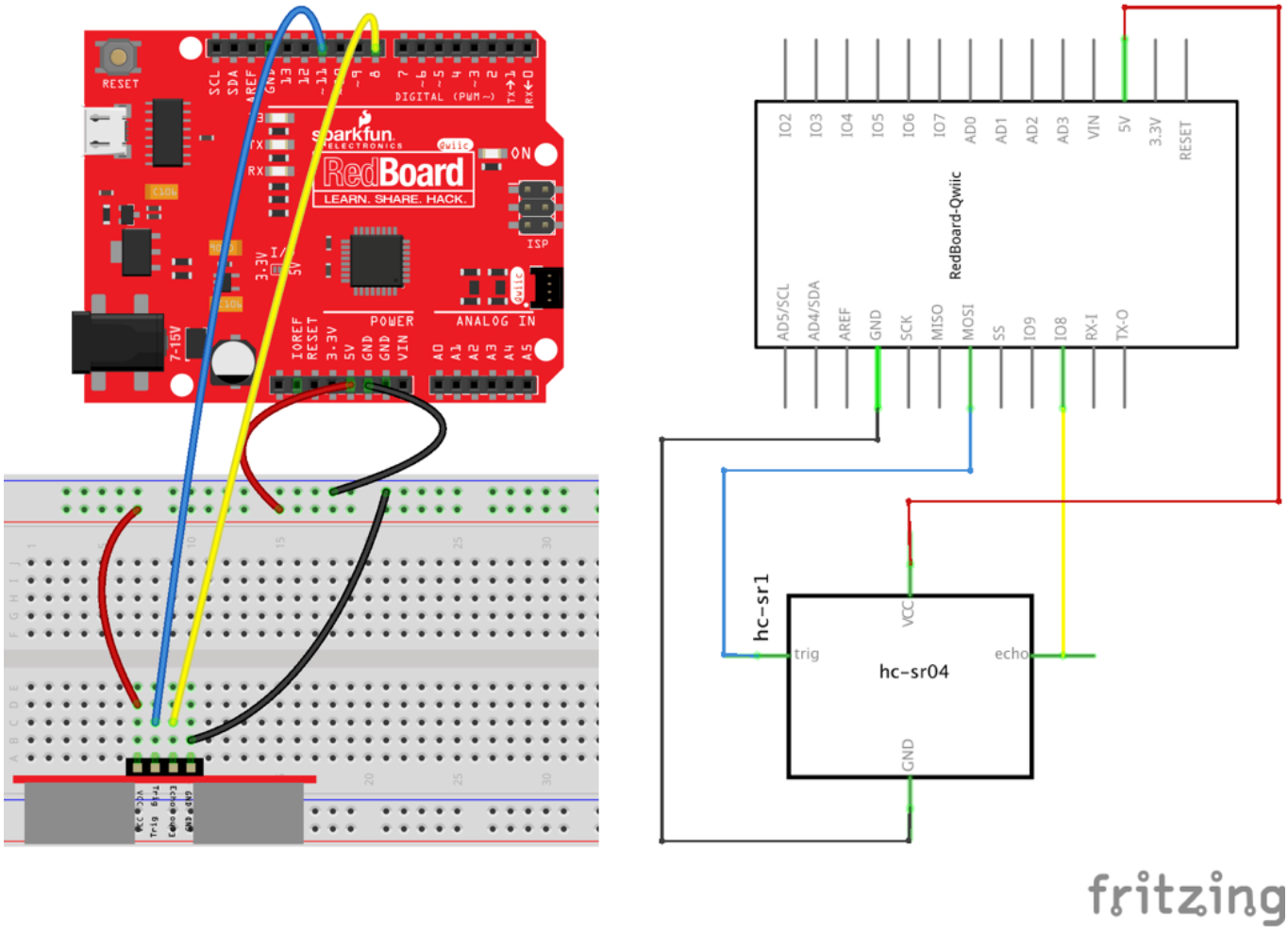
Note the pins labeled VCC (5V), Trig (trigger), Echo, and GND. Build the circuit below. The Ultrasonic sensor should face an open side, away from the Arduino

Ultrasonic Distance Measurement and Serial Monitor Display Software

Open the program, Lab7_distance_serial_display.ino, and look at it. Note in setup the command: Serial.begin(9600); This establishes two-way communications between the Arduino and the IDE, which continues as long as the USB cable is connects the Arduino to the computer. The Arduino can send information to the IDE which will be displayed on a serial monitor. To see this monitor, on the IDE tools→serial monitor move the window as you please, and make sure that the communication speed you declare in the program (9600 in this case) matches the speed in the monitor (9600 baud)

In the main loop, you can find these two commands:

Serial.print(distance);



Serial.println(" inches");

These will cause the IDE to display the value of the variable "distance" and the text " inches" then start a new line. Once you upload the program, the display will start running.

To get the distance to an object in front of the ultrasonic sensor, it sends an 8-cycle square-wave burst at a frequency of 40kHz. That's high enough that we cannot hear it, but your pet dog, cat and opossum might be bothered by it. The Ultrasonic sensor starts the burst when it receives a 10 microsecond LOW-HIGH-LOW

pulse. You can see that in the program. Arduino has a built-in function to time how long it takes for the pulse to bounce back to the ultrasonic sensor, pulseIn(pin, pulse type). This is used in the distance calculation by the formula:

$$distance = \frac{pulse\ return\ time}{2} \cdot speed\ of\ sound = \frac{pulseIn(8, HIGH)}{2} \cdot 0.01242 \frac{inches}{microsecond}$$

```
distance=pulseIn(8,HIGH)/2*0.01242;
```

(the speed of sound at 6,200 feet altitude is about 1035 ft/sec or 0.01242 inch/microsecond. For sea level use 0.013548 inch/microsecond).

Experiment with your ultrasonic sensor. Point it at various objects, near and far. It should work well in the range of about 5 inches to 30 inches. It may need a flat object at least the size of a small book to sense well.

To receive credit for this lab, you *must* get the sign-off before the due date (see Canvas), during class time or with a TA during their hours. The instructor will not signoff outside of class hours. Partial credit is allowed.