Aberystwyth Robotics Club

Magician Chassis

Programming Instructions

Author: Tomos Fearn

V1.0 / Release

18th October 2017

Motor Instructions

The motors on the Magician Chassis are controlled by a dual h-bridge motor controller, these are attached to the following pins:
These will need to be defined at the top of the sketch, above void setup().

```
// Declaring Variables
byte leftForward = 4;
byte leftReverse = 5;
byte rightForward = 9;
byte rightReverse = 10;
```

You will need to define the motors inside the setup function as output devices:

```
void setup(){
  pinMode(leftForward, OUTPUT);
  pinMode(leftReverse, OUTPUT);
  pinMode(rightForward, OUTPUT);
  pinMode(rightReverse, OUTPUT);
}
```

Functions()

You need to create 5 functions which give instructions to the robot, create the functions below:
“A function is to divide up your code into chunks, this will make it easier to create instructions to the robot and prevents code duplication!”

- robotForwards()
- robotLeft()
- robotRight()
- robotReverse()
- robotStop()

This can be done by sending digital ’HIGH’ and ’LOW’ to the motors, such as the example below: the left code shows how to control speed whereas the right is full speed.

```
void robotForwards(){
    analogWrite(leftForward, 255);
    analogWrite(rightForward, 255);
    analogWrite(leftReverse, 0);
    analogWrite(rightReverse, 0);
}
void robotLeft(){
    analogWrite(leftForward, 0);
    analogWrite(rightForward, 255);
    analogWrite(leftReverse, 255);
    analogWrite(rightReverse, 0);
}
```

```
void robotForwards() {
    digitalWrite(leftForward, HIGH);
    digitalWrite(rightForward, HIGH);
    digitalWrite(leftReverse, LOW);
    digitalWrite(rightReverse, LOW);
}
void robotLeft() {
    digitalWrite(leftForward, LOW);
    digitalWrite(rightForward, HIGH);
    digitalWrite(leftReverse, HIGH);
    digitalWrite(rightReverse, LOW);
}
```

Once you create the instructions in the functions you can call the function in void loop(), try all the functions you have created and make the robot move around:

```
void loop(){
  robotForwards();
}
```

Light Dependant Resistors Instructions

Abstract

In order to detect the intensity of light or darkness, we use a sensor called an LDR (Light Dependent Resistor). The LDR is a special type of resistor which allows higher voltages to pass through it (low resistance) whenever there is a high intensity of light, and passes a low voltage (high resistance) whenever it is dark.

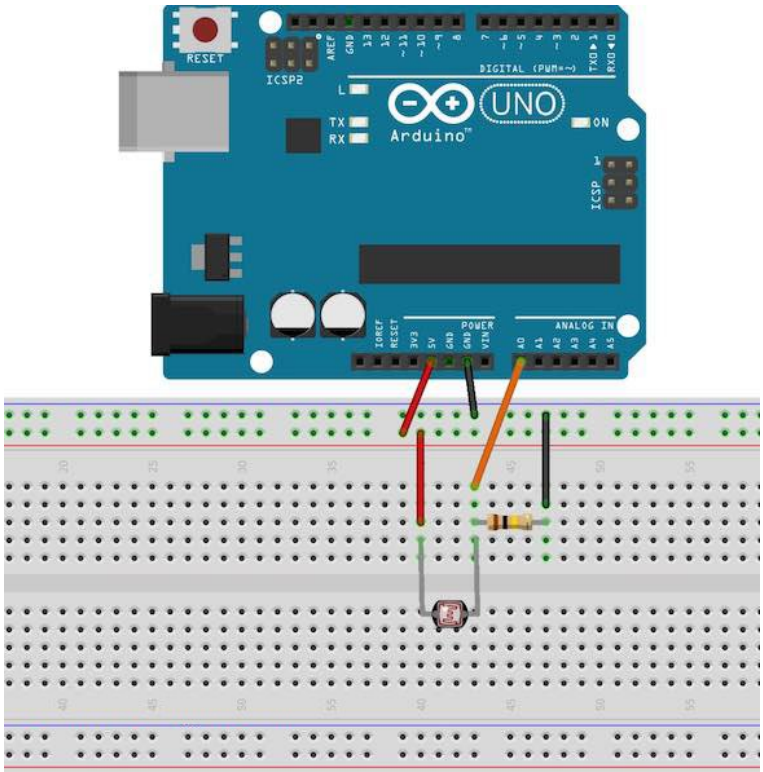
1. Hardware Required

- Arduino or an Arduino clone board (Freeduino), or make your own custom Arduino board.
- LDR (you can buy it online or from a local electronics store)
- Connecting wires
- 100K resistor

2. Circuit

First of all, you need to connect the LDR to the analog input pin 0 on the Arduino. You have to use a voltage divider configuration to do this. The connection diagram for the Arduino is as given below.

One leg of the LDR is connected to VCC (5V) on the Arduino and the other to the analog pin 0 on the Arduino. A 100K resistor is also connected to the same leg and grounded.



3. Code

After you build the circuit plug your Arduino or Genuino board into your computer, start the Arduino Software (IDE) and enter the code below. In the program below, the very first thing that you do will in the setup function is to begin serial communications, at 9600 bits of data per second, between your board and your computer with the line:

```
pinMode(13, OUTPUT);
```

Next, initialize digital pin 2, the pin that will read the output from your button, as an input:

```
pinMode(2,INPUT);
```

This supplies 5 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

```
digitalWrite(13, LOW);
```

Now that your setup has been completed, move into the main loop of your code. When your button is pressed, 5 volts will freely flow through your circuit, and when it is not pressed, the input pin will be connected to ground through the 10k ohm resistor. This is a digital input, meaning that the switch can only be in either an on state (seen by your Arduino as a "1", or HIGH) or an off state (seen by your Arduino as a "0", or LOW), with nothing in between.

The first thing you need to do in the main loop of your program is to establish a variable to hold the information coming in from your switch. Since the information coming in from the switch will be either a "1" or a "0", you can use an int datatype. Call this variable sensorValue, and set it to equal whatever is being read on digital pin 2. You can accomplish all this with just one line of code:

```
int sensorValue = digitalRead(2);
```

Once the bord has read the input, make it print this information back to the computer as a decimal value. You can do this with the command Serial.println() in our last line of code:

```
Serial.println(sensorValue);
```

Now, when you open your Serial Monitor in the Arduino Software (IDE), you will see a stream of "0"s if your switch is open, or "1"s if your switch is closed.

```
int pushButton = 2; // digital pin 2 has a pushbutton attached to it.

void setup() {
  Serial.begin(9600); // initialize serial communication at 9600 bits per
second:
  pinMode(pushButton, INPUT); // make the pushbutton's pin an input:
}

void loop() { // the loop function runs over and over again forever
  int buttonState = digitalRead(pushButton); // read the input pin:
  Serial.println(buttonState); // print out the state of the button:
  delay(1); // delay in between reads for stability
}
```

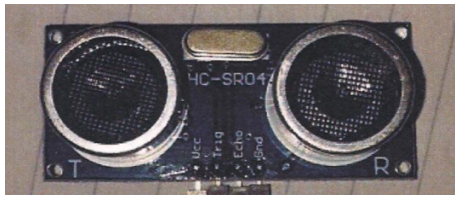
Ultrasonic Instructions

Abstract

This example shows you how to find the distance from an obstacle using ultrasonic sensors, for now we will return the values from the ultrasonic to the serial monitor on the Arduino.

1. Hardware Required

- Arduino or Genuino Board
- HC-SR04 Ultrasonic
- Jumper Cables
- Optional breadboard



2. Circuit

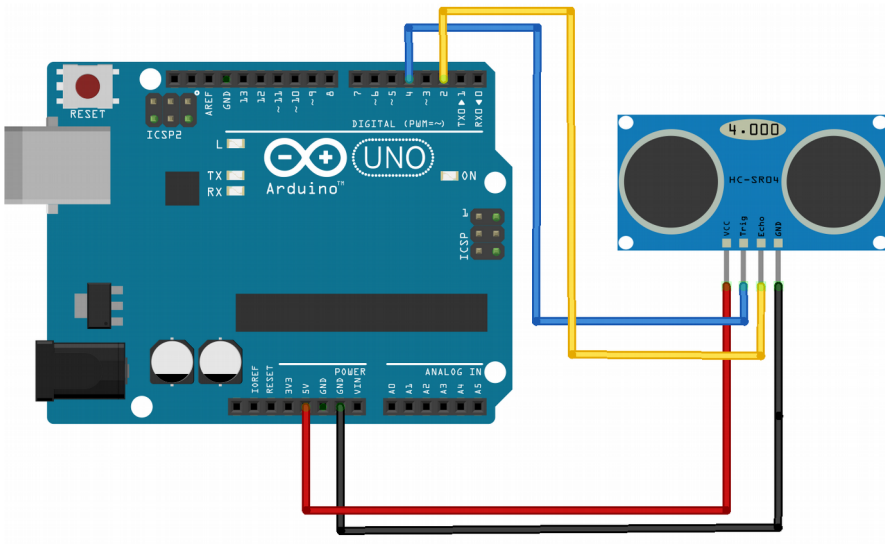
Ultrasonics work by sending out pulses of sound, and then counting how long it takes for those pulses to come back. So a Ultrasonic sensor actually has two parts: a transmitter and a receiver

If you look at the sensor it has two round bits, labelled T for transmitter and R for receiver. (In electronics things which both transmit and receive signals are often called “Transceivers” so tgus is really a Ultrasonic transceiver.)

The Ultrasonic sensor has 4 wires labelled Vcc, Trig, Echo and Gnd.

These are for the voltage, the trigger of the sonar, the echo detection signal, and the ground.

- The first of these (voltage, or Vcc) needs to be wired up to the 5v output of your Arduino.
- The trigger or Trig needs to be wired up to a data pin on the Arduino - let’s wire it up to number 2.
- The echo detection signal also goes to a data pin on the Arduino – let’s wire it up to number 3.



3. Sending out a pulse and timing it coming back

Remember that all Arduino programs have two parts: the setup which happens once, and is used to set up anything that you need to use in the program. This is where you tell the Arduino which components are wired up to which pin, and stuff like that. There is also a loop, and the loop is repeated again and again until the Arduino is turned off (or runs out of battery power).

In the setup function here we need to tell the Arduino which pin has the Trig connection, and which pin has the Echo connection.

We are going to use the trigger pin to send a message to the sensor from the Arduino (so that is an OUTPUT pin), but we are going to use the Echo pin to read a message from the sensor (so that is an INPUT pin.)

4. Code

After you build the circuit plug your Arduino or Genuino board into your computer, start the Arduino Software (IDE) and enter the code below.

```
void setup () {
  Serial.begin(9600);
  pinMode(2, OUTPUT);
  pinMode(3, INPUT);
}
```

As well as telling the program what's connected to what pin, it also sets up the serial monitor so we are ready to start monitoring stuff.

Now we have set up the program we need to trigger the sensor to send out a signal, then read the signal coming back from the sensor. We can do this as follows:

```
void loop () {
  long duration;
  digitalWrite(2, HIGH);
  delayMicroseconds(10);
  digitalWrite(2, LOW);
  duration = pulseIn(3, HIGH);
  Serial.print(duration);
  Serial.println(" : duration ");
}
```

There is a lot going on in this program.

- First we setup a variable called duration to hikd the length of time between the sound going out and the sound bouncing back.
- Gben we trigger a quick pulse of sound (by writing HIGH to the trigger pin) - we only do this for a tiny amount of time (10 microseconds) before turning it off again by writing LOW to the trigger pin.
- We then capture the time from the pulse being sent out to coming back by looking for a HIGH pulseIn on the echo pin.
- Finally we print the duration out to the serial monitor.

Type this program in, test it by clicking the tick, then put it on the Arduino using the arrow button.

To open the serial monitor, click on the button that looks a bit like a magnifying glass in the top right corner of the Arduino IDE window (it will say "Serial Monitor" when you hover over it). That will open up a new window which contains the stuff you have printed to Serial.

You should see something like this:

```
386: duration
417: duration
381: duration
```

These are measurements of how long it took (in microseconds) for the sound

to bounce off a thing in the world and then back to the ultrasonic sensor.

5. Going from a measurement of time to a measurement of distance

Set up a barrier near your computer, and get hold of a ruler.

Measure the distance from the sensor to the barrier, and fill in a table of measurements from the serial monitor.

If the barrier is 10cm from the sensor, how long does it take the sound to travel?

If the barrier is 10cm from the sensor, how long does it take the sound to travel?

Take some more measurements. Can you come up with a way of converting “sound-travel-time” into centimetres?

If you can come up with a useful conversion system, can you then get that into your program - that is, can you work out how to get your program to give an output in centimetres? Give it a go, and then test your ultrasonic ruler against a real ruler.

Behaviours

You will need to make your magician chassis do the following behaviours

1. Control the wheels using a source of light (such as a torch).
2. Avoid obstacles using the Ultrasonic sensor.
3. Should be able to transport a ball using all hardware on board the robot.