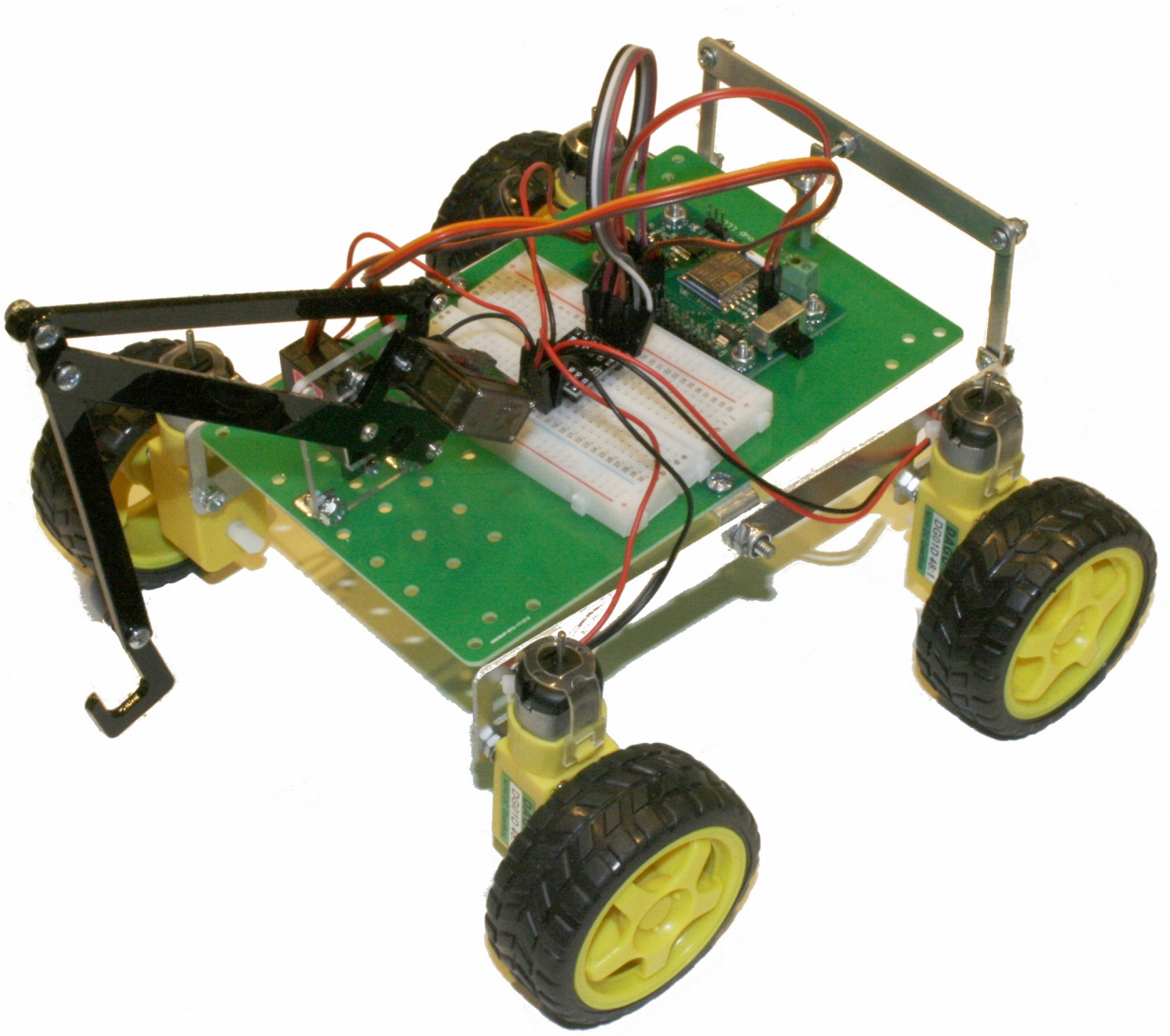
*Sten**BOT*** Robot Kit



Legal Stuff

- Stensat Group LLC assumes no responsibility and/or liability for the use of the kit and documentation.
- There is a 90 day warranty for the Quad-Bot kit against component defects. Damage caused by the user or owner is not covered.
    - Warranty does not cover such things as over tightening nuts on standoffs to the point of breaking off the standoff threads, breaking wires off the motors, causing shorts to damage components, powering the motor driver backwards, plugging the power input into an AC outlet, applying more than 9 volts to the power input, dropping the kit, kicking the kit, throwing the kit in fits of rage, unforeseen damage caused by the user/owner or any other method of destruction.
- If you do cause damage, we can sell you replacement parts or you can get most replacement parts from online hardware distributors.
- This document can be copied and printed and used by individuals who bought the kit, classroom use, summer camp use, and anywhere the kit is used. Stealing and using this document for profit is not allowed.
- If you need to contact us, go to www.stensat.org and click on contact us.

References

- www.arduino.cc
- esp8266.githun.io/Arduino/versions/2.3.0

Robot Sensing

- This section, you learn about using sensors to control the robots movements.
- Make sure the power switch is off and the USB cable is disconnected before wiring up the next circuit.

Photo Cell

- Disconnect the USB cable and make sure the power switch is off.
- The photo cell is a light sensitive device that changes its resistance based on light intensity.
- The photocell can be used in a simple voltage divider circuit with another resistor. The resistor is 100Kohms.
- The photo resistor will have a resistance ranging from 1 Mohm in darkness to 100 ohms in bright light.
- Install the photo cell and 100 K resistor on the solderless bread board away from the motor driver. Make sure the photo cell and resistor are connected.
- Connect the free end of the resistor to V3 at the analog connector.
- Connect the free end of the photo cell to GND.
- Connect the resistor and photo cell connection to pin ADC of the analog connector.

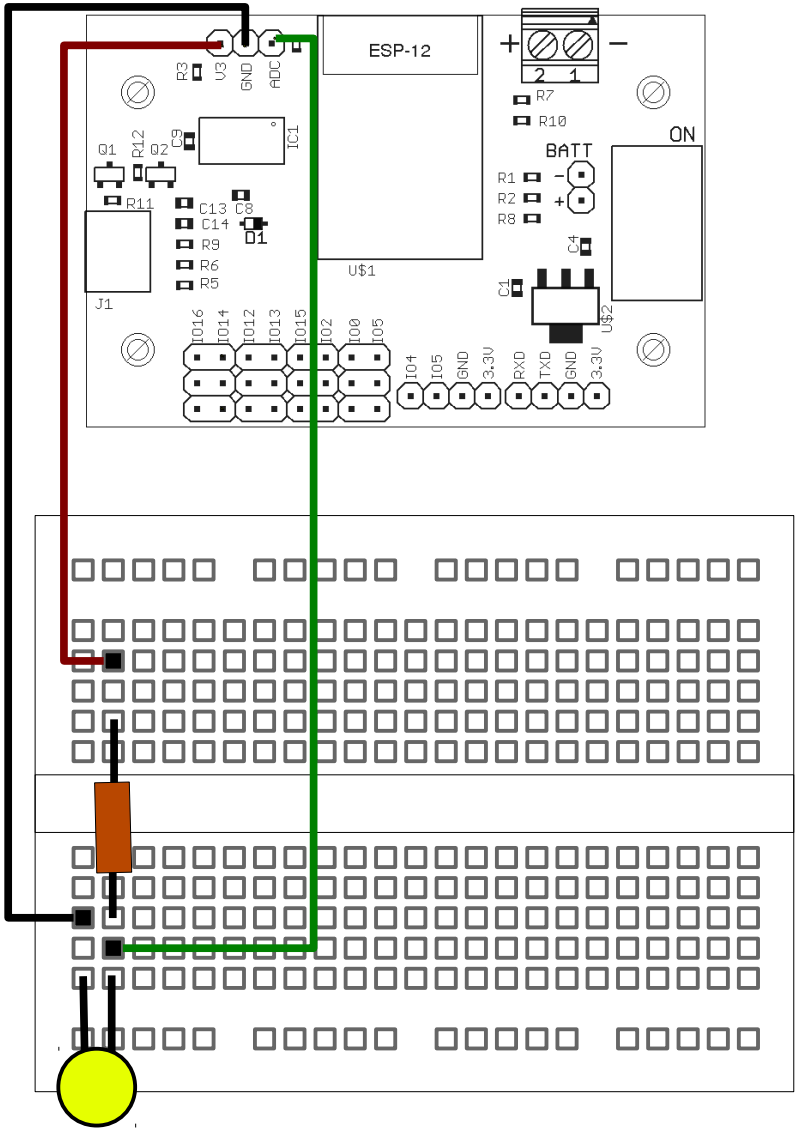


Photo Cell Program

- The program to the right will get an ADC value from analog port **A0**.
- Create a new program and enter the code.
- To measure the voltage, the function **`analogRead(port)`** is used.
- One analog port is available and it is called **A0**. The voltage range on the analog port is 0 to 1 volt.
- Once the ADC value is read, it can be converted to a voltage value. The code to the right shows the equation which can be used for all the analog ports.
- The **`Serial.println()`** function that displays the volts, includes a numeric argument which specifies the number of decimal places.
- Save the program to a new file.

```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0;
  Serial.println(volts,2);
  delay(200);
}
```

Conditional Programming

- Before the **delay()** function, add the **if()** statement in bold.
- If the equation in the parentheses is true then the code in the brackets after the **if()** statement will be executed.
- Run the code and see if it works. If the area is very bright, the comparison value **0.8** can be reduced.

```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0;
  Serial.println(volts,2);
  if(volts > 0.8) {
    Serial.println("It is dark");
  }
  delay(200);
}
```

Conditional Programming

- Now, add an LED to pin 5. Don't forget to include the 270 ohm resistor.
- Modify the code to look like the one at the right.
- This time, the LED is turned on when it is dark.
- Notice the **else** statement added after the bracket. If the equation is false, then the code in the brackets after the **else** statement is executed.

```
void setup()
{
  pinMode(16,OUTPUT);
  Serial.begin(115200);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0;
  Serial.println(volts,2);
  if(volts > 0.8) {
    Serial.println("It is dark");
    digitalWrite(16,HIGH);
  } else {
    digitalWrite(16,LOW);
  }
  delay(200);
}
```

Quiz Time

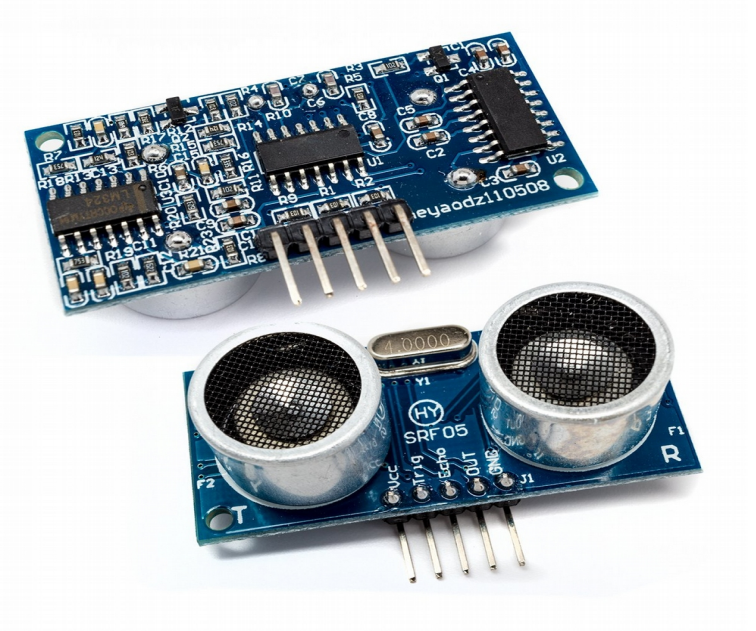
- Now that you know how the photo cell works and how to control LEDs, write a program to turn on an LED when the room becomes dark.
- Write a program to detect three levels of light and turn on the appropriate LED. You can use pin 4 for the second LED.
  - If the room is completely dark, no LEDs are lit.
  - If the room light is dim, turn on the red LED.
  - If the room is bright, turn on the green LED.
  - It is up to you to select the thresholds.

Light Seeking Program

- The photo cell can be used to have the robot chase after a light source.
- Write a program to have the robot look for a bright light source. If one is not detected, have the robot turn in place. When it detects a light source, have the robot move forward toward the light source.
- Bend the photo cell to face frontward. Use a flash light and shine it toward the robot. You may need to adjust the value in the **if()** statement to make it work.

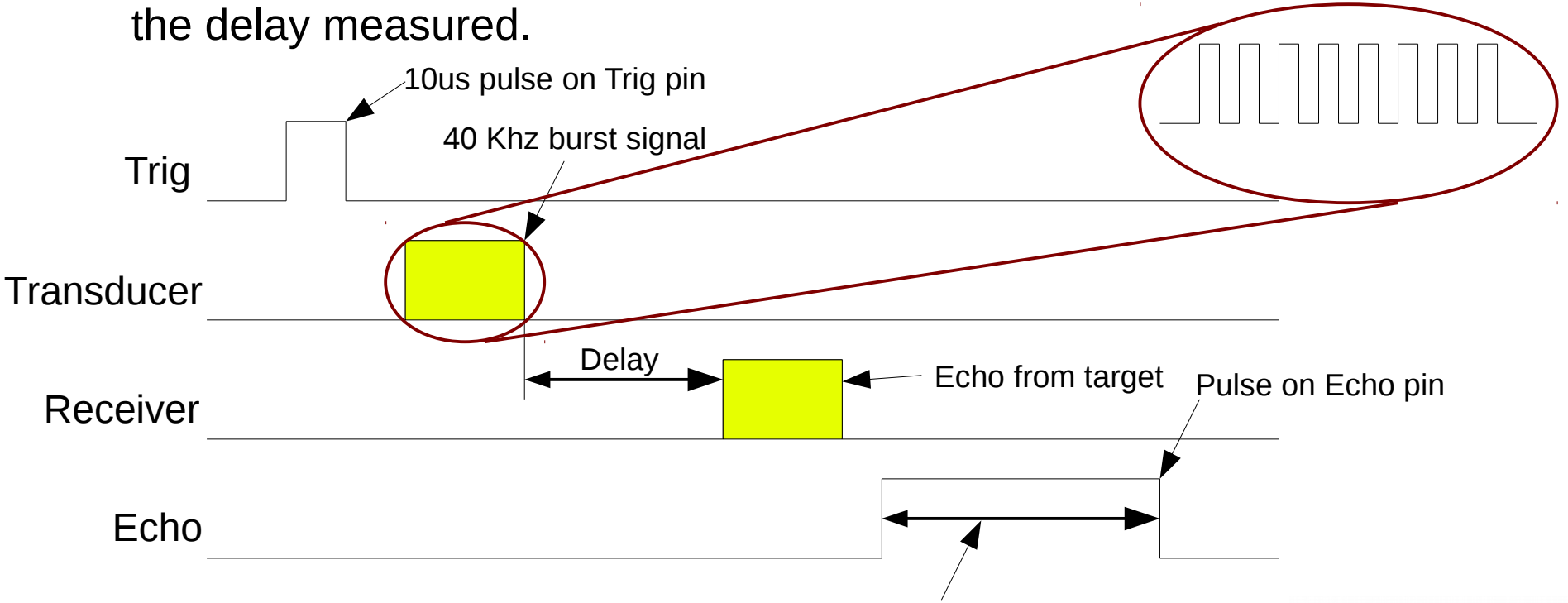
Sensing the Environment

- To detect things in the environment for purpose of collision avoidance, an ultrasonic range sensor will be added to the robot.
- This sensor sends out a burst of audio signal at 40 KHz and detects the echo.
- The processor needs to measure the time it takes for the echo to return.
- This sensor has four pins
  - Ground
  - 5 Volt power input
  - Trigger
  - Echo



Ultrasonic Range Sensor Operation

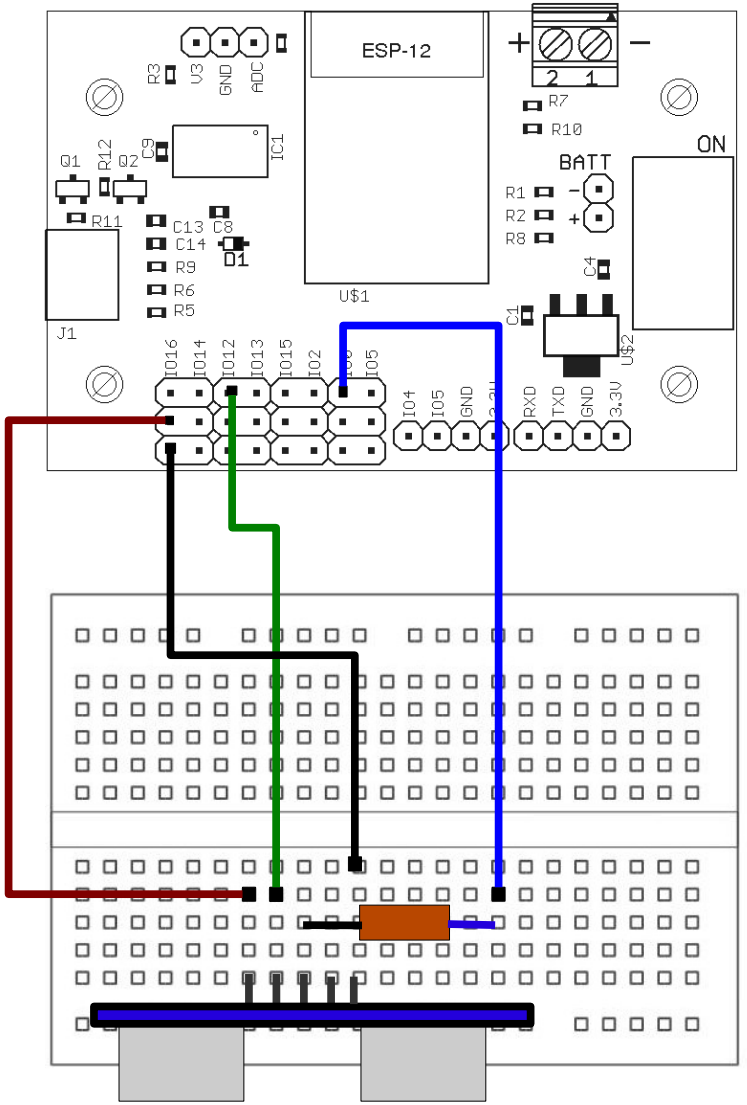
- The ultrasonic range sensor operates in a specific sequence.
- It waits for a trigger signal. The trigger is a 10us pulse. Once the trigger is detected, the sensor generates a short signal at 40 KHz.
- It then waits for an echo and measures the time from sending the short burst to receiving the echo.
- The sensor then generates a pulse on the echo with a length proportional to the delay measured.



distance = pulse width (us) / 58

Ultrasonic Range Sensor

- Disconnect the USB cable and make sure the power switch is off and remove the light sensor circuit and install the ultrasonic range sensor in the same area as shown to the right.
- Connect the VCC pin from the sensor to the 5V pin on digital pin 16.
- Connect the GND pin from the sensor to the GND pin on digital pin 16.
- Connect the TRIG pin from the sensor to digital pin 12.
- Connect a 1K resistor (brown-black-red) from the ECHO pin to a spot on the bread board. Then connect the other end of the resistor to digital pin 0.



Ultrasonic Sensor

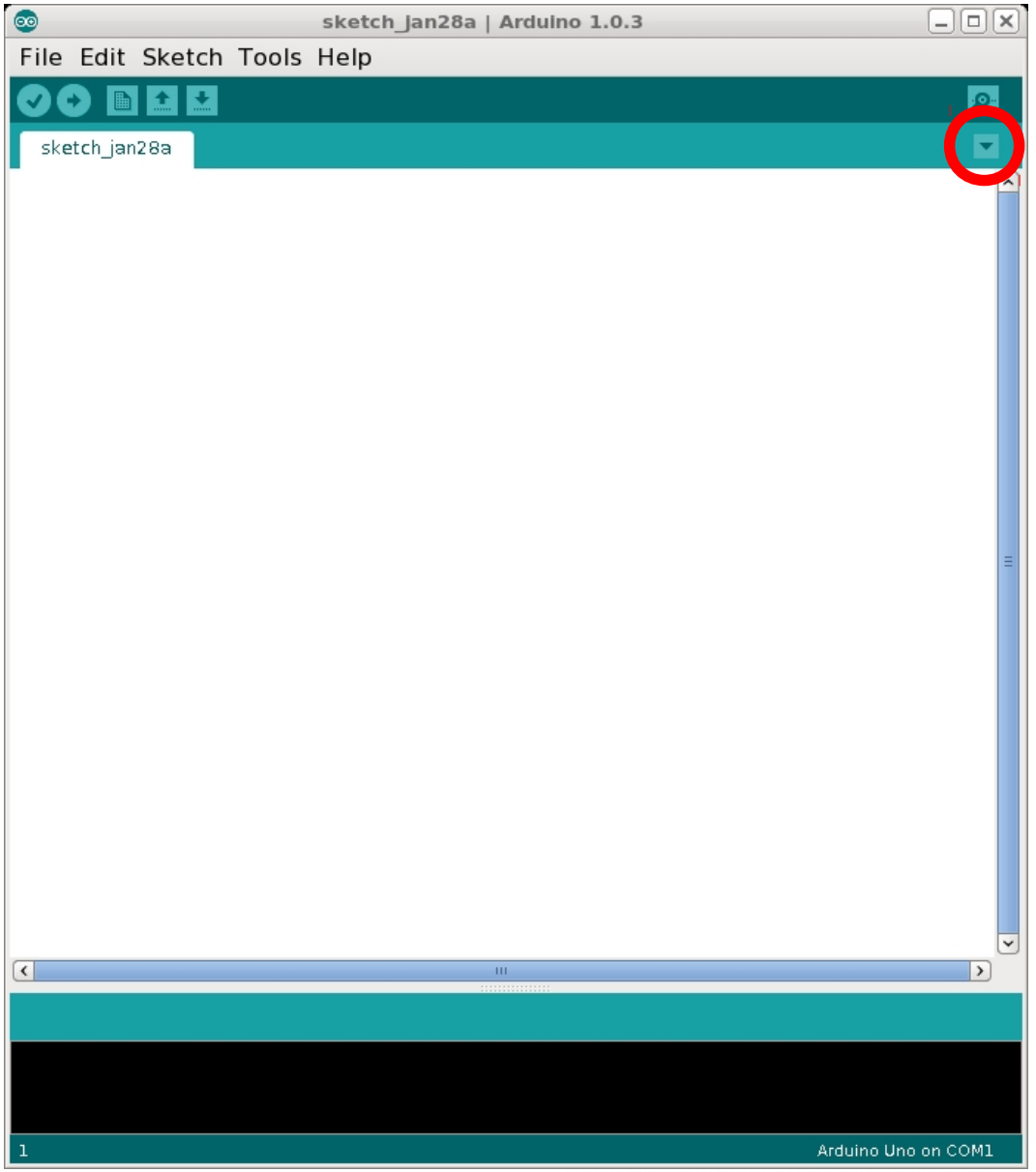
- The ultrasonic sensor has two signals, trigger and echo.
- A pulse is sent to the trigger and then the processor is to time when the echo returns.
- This requires two digital pins, one configured as an output and the other as an input. A new command that will be used is called **`pulseIn()`**. This measures the time it takes a pulse to occur in microseconds. Try the program to the right.
- The results are in centimeters.
- Create a new program and enter the code to the right. Save the program and upload it.

```
void setup()
{
    Serial.begin(115200);
    pinMode(0,INPUT);
    pinMode(12,OUTPUT);
}

void loop()
{
    unsigned long range;
    unsigned long distance;
    digitalWrite(12,LOW);
    delayMicroseconds(2);
    digitalWrite(12,HIGH);
    delayMicroseconds(10);
    digitalWrite(12,LOW);
    range = pulseIn(0,HIGH);
    distance = range/58;
    Serial.println(distance);
    delay(500);
}
```

Creating a Separate Function File

- Click on the down arrow to the right where circled in red.
  - A menu will open. Select “New Tab”
  - Below, it will ask for a name. Enter 'motion'
  - Click 'OK'
  - A new tab is created called 'ultrasound'



Ultrasonic Sensor

- The ultrasonic sensor has two signals, trigger and echo.
- A pulse is sent to the trigger and then the processor is to time when the echo returns.
- This requires two digital pins, one configured as an output and the other as an input. A new command that will be used is called **`pulseIn()`**. This measures the time it takes a pulse to occur in microseconds. Try the program to the right.
- The results are in centimeters.
- Create a new program and enter the code to the right. Save the program and upload it.

```
void setup()
{
    Serial.begin(115200);
    pinMode(0,INPUT);
    pinMode(12,OUTPUT);
}

void loop()
{
    unsigned long distance;
    digitalWrite(12,LOW);
    delayMicroseconds(2);
    digitalWrite(12,HIGH);
    delayMicroseconds(10);
    digitalWrite(12,LOW);
    distance = pulseIn(0,HIGH);
    distance = distance/58;
    Serial.println(distance);
    delay(500);
}
```

Creating a Separate Function File

- Select **Save As...** under the **File** menu.
- Give the program the name **ultrasound_f**.
- Next, delete the function setup(). It is highlighted in red.

```
void setup()
{
    Serial.begin(115200);
    pinMode(0,INPUT);
    pinMode(12,OUTPUT);
}

void loop()
{
    unsigned long distance;
    digitalWrite(12,LOW);
    delayMicroseconds(2);
    digitalWrite(12,HIGH);
    delayMicroseconds(10);
    digitalWrite(12,LOW);
    distance = pulseIn(0,HIGH);
    distance = distance/58;
    Serial.println(distance);
    delay(500);
}
```

Creating a Separate Function File

- Now, change **void loop()** to **unsigned long ultrasound().**
- Next, delete the last two lines in the function highlighted in red.

```
unsigned long ultrasound()
{
    unsigned long distance;
    digitalWrite(12,LOW);
    delayMicroseconds(2);
    digitalWrite(12,HIGH);
    delayMicroseconds(10);
    digitalWrite(12,LOW);
    distance = pulseIn(0,HIGH);
    distance = distance/58;
    Serial.println(distance);
    delay(500);
}
```

Creating a Separate Function File

- Insert the **return** command at the end of the function.
- Select **Save** from the **File** menu.
- You now created a function that can be added to any future program.

```
unsigned long ultrasound()
{
    unsigned long distance;
    digitalWrite(12,LOW);
    delayMicroseconds(2);
    digitalWrite(12,HIGH);
    delayMicroseconds(10);
    digitalWrite(12,LOW);
    distance = pulseIn(0,HIGH);
    distance = distance/58;
    return(distance);
}
```

Creating a Separate Function File

- Click on the first tab.
- Enter the program to the right.
- Save the program with the name **ranging**. You will notice the first tab is now named **ranging**.
- Under the **Sketch** menu, select **Add File...**
- Select the ultrasound_f program. You have to go into the ultrasound_f folder to select the file.
- A new tab will appear with the ultrasound function.
- Compile the progam and upload it.

```
void setup()
{
    Serial.begin(115200);
    pinMode(0,INPUT);
    pinMode(12,OUTPUT);
}

void loop()
{
    unsigned long distance;
    distance = ultrasound();
    Serial.println(distance/58);
    delay(500);
}
```

Conditional Programming

- Now it is time to use the ultrasonic sensor to do collision avoidance.
- The 'if' command will be used to test if the robot will collide with an object.
- The format for the if statement is shown to the right.
- Multiple statements can be inserted between the brackets and will be executed if the condition is true.
- To test for equals, use '=='
- else allows two sets of codes to be executed depending on the condition.

```
if(a < c) {
    execute code here
}

if(a == c) {
    execute this code
}

if(a > c) {
    execute this code
} else {
    otherwise execute this code
}
```

Collision Avoidance Program

- The program on the next page will use the code used to control the motors, the ultrasonic function, and the conditional command.
- Put together, the program will keep the robot from bumping into anything.
- Enter the code on the next page. The code should be written in a single file. The code is split on the next page since it wouldn't fit in a single column.
- You will notice a **`delay()`** at the end of the **`loop()`** function. This is needed because the ultrasonic range sensor cannot be operated too fast. Incorrect results will occur if the loop runs too fast.
- Test it and see if you need to tweak the timing for going reverse and turning.
- Don't forget to include the motion file by adding the file.
- Save the program and then upload it.
- Change the code to turn a different direction.

Collision Avoidance Program

```
long ultrasonic()
{
     digitalWrite(12,LOW);
     delayMicroseconds(2);
     digitalWrite(12,HIGH);
     delayMicroseconds(10);
     digitalWrite(12,LOW);
     long distance = pulseIn(0,HIGH);
     if(distance == 0)
          return(1000);
     distance = distance/58;
     return(distance);
}

void setup()
{
     pinMode(0,INPUT);
     pinMode(15,OUTPUT);
     pinMode(16,OUTPUT);
     pinMode(14,OUTPUT);
     pinMode(13,OUTPUT);
     pinMode(12,OUTPUT);
}
```

```
void loop()
{
long distance;
     forward();
     distance = ultrasonic();
     if(distance < 10) {
          reverse();
          delay(1000);
          left();
          delay(700);
          halt();
     }
     delay(50);
}
```

Obstacle Course Time

- Now for the fun part. Modify and expand the program to go through the obstacle course shown below. The large square represent 2 foot grids. The red rectangles represent a barrier that can be detected with the ultrasonic range sensor. Set up some barriers out of any solid material. Card board boxes, poster paper, or other large materials will work.
- Use the ultrasonic range sensor to avoid crashing into the barriers and turns the right direction every time a barrier is detected.
- Hint, use the collision avoidance program and expand it so that it will complete the maze. This requires the robot to back up and turn in specific directions at specific points of the maze.