
CRC
CREATIVE
ROBOTICS
CLUB

WELCOME TO THE

CREATIVE ROBOTICS CLUB

WHAT DO WE DO AT THE **CREATIVE ROBOTICS CLUB?**

We learn how to use electricity,
robotics and code to make things

We make art, design, or social robotics
- we support all disciplines

We reuse and repurpose where we can

We have fun

HOW DO WE RUN **CREATIVE ROBOTICS CLUB?**

WEEKS 2 - 5: Skill acquisition
We will learn new skills, try new ideas, grow our knowledge each week

WEEKS 7 - 10: Project support
Have the things you've learned in Weeks 2 -5 got you itching to make something? Do you have assignments that need electronics or programming support?
We are here to help.

WE ARE OPEN TO YOUR FEEDBACK!

Are there things you want us to talk about?

A different way of running you think will work?

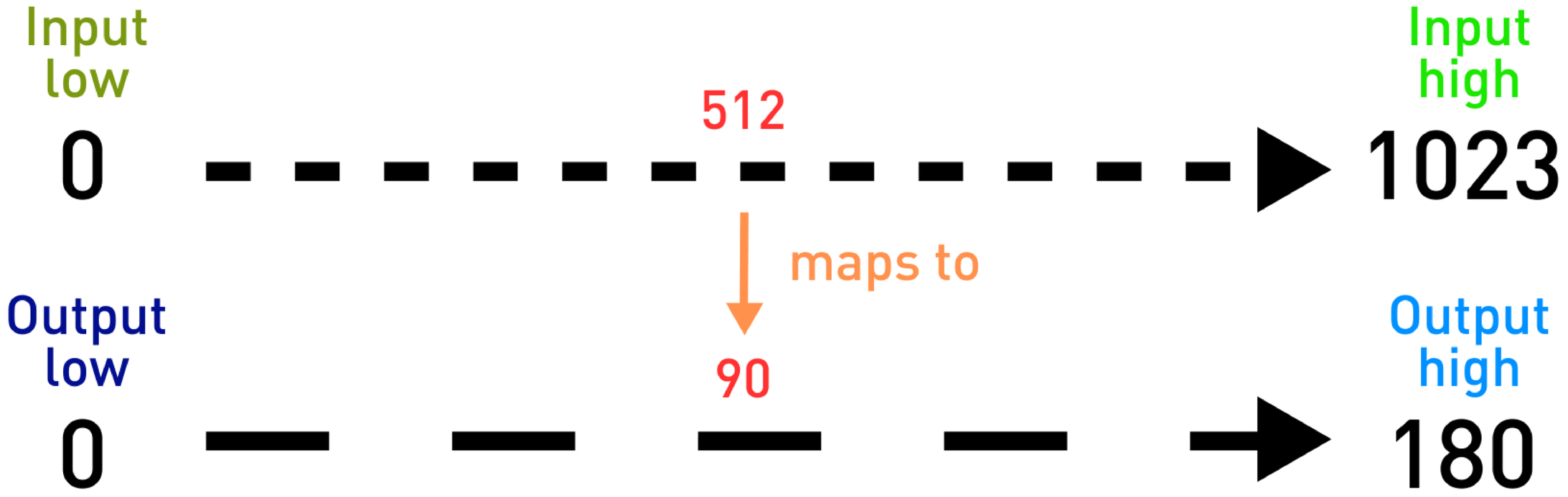
Skills you want to share?

We are a club for students, and we welcome your suggestions and input

BUT FIRST LETS TALK ABOUT...

LAST WEEK

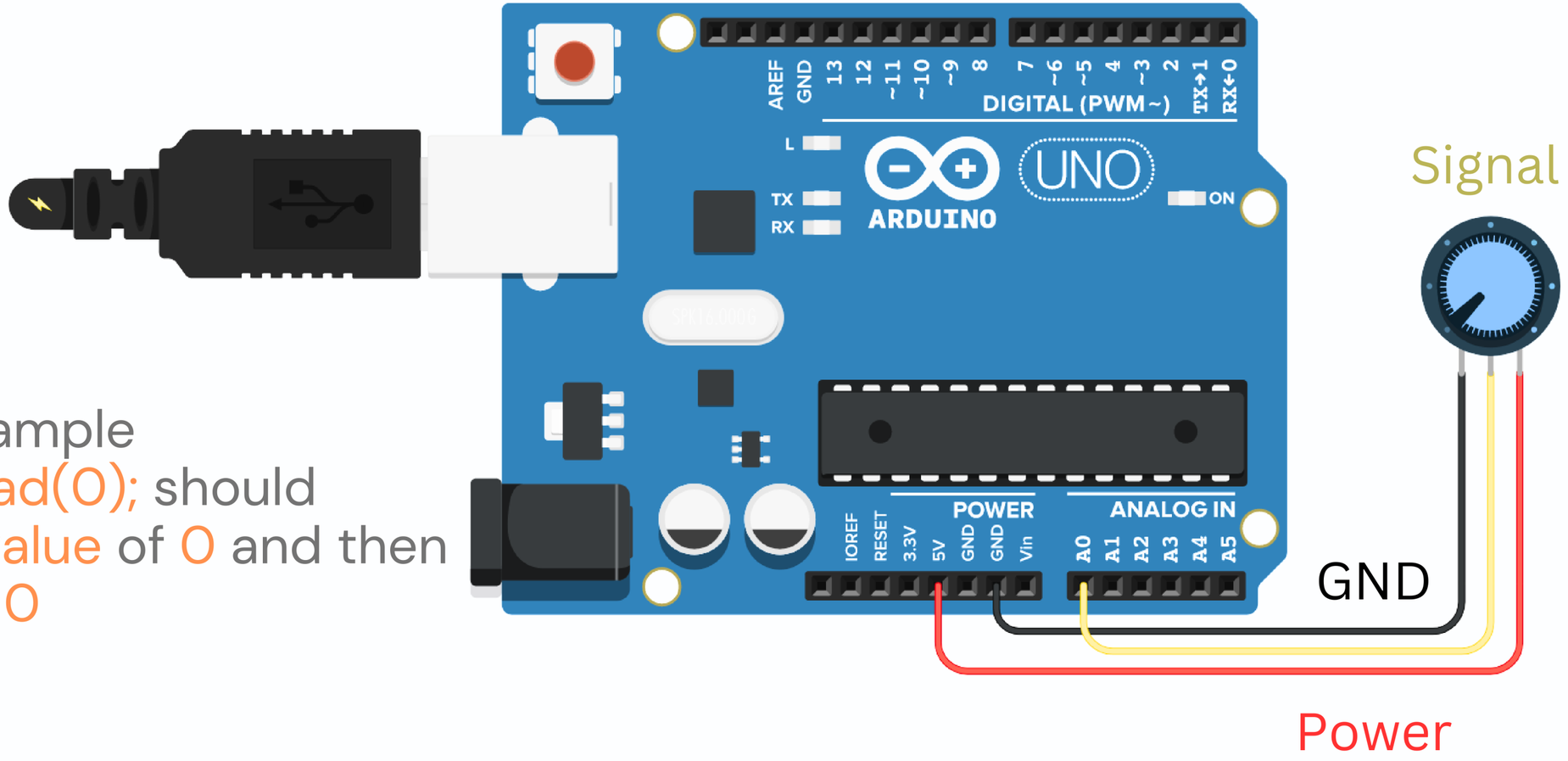
map( ) lets you take one range of values and scale them to match another




```
map(value, inputLow, inputHigh, outputLow, outputHigh);
```

ARDUINO

In this example analogRead(0); should return a value of 0 and then map it to 0



NOW WHEN YOU RUN THIS CODE...

A list of values will start appearing in the Serial Monitor.

Now we know what's going on!

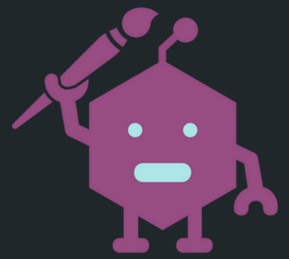
We can use this information to drive interaction in our system.

sketch_mar2a.ino

```
1   void setup() {
2     Serial.begin(115200); //Start the Serial monitor
3   }
4
5   void loop() {
6     //Read Analong In pin 0
7     int value = analogRead(A0);
8     //Map the reading to a number between 0-180
9     value = map(value, 0, 1023, 0, 180);
10    //Add text so we know what we're looking at
11    Serial.print("Value = ");
12    //Write the value
13    Serial.println(value);
14    //Wait 15ms and do it again
15    delay(15);
16  }
```

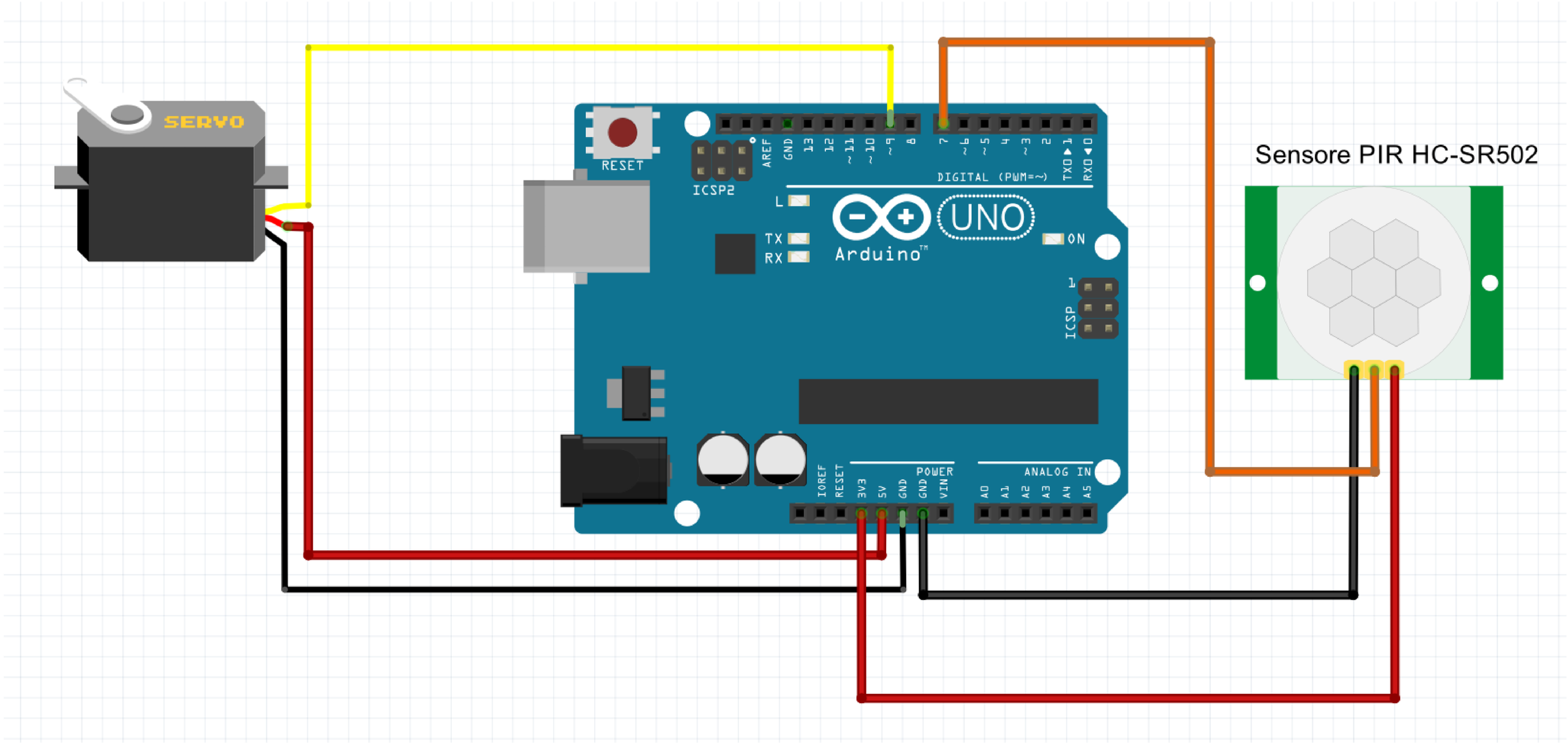
Serial Monitor ×

Message (Enter to send message to 'Arduino U... | New Line | 115200 baud



S = SIGNAL V = VOLTAGE and G = GND

So, now we know how to wire this sensor up!



WE ALSO LEARNED THAT PIR SENSORS CAN SUCK SOMETIMES

WHERE CAN I FIND THE SLIDES FROM **CREATIVE ROBOTICS CLUB?**

github.com/instasquid/creativeroboticsclub

WHAT ARE WE DOING TODAY AT THE **CREATIVE ROBOTICS CLUB?**

ULTRASONIC DISTANCE SENSORS
REUSING CODE
SMOOTHING DATA
IF STATEMENTS (AGAIN)
WHAT'S NEXT?

ARDUINO

So lets talk bout other types of sensors.

There Arduino has many different types of sensor for all sorts of applications: temperature, flexibility, sound, to check for the presence of a magnetic field, to check light levels, flame sensors, and more.

Last week we looked at the PIR sensor to detect movement, lets take a quick look at another one today.

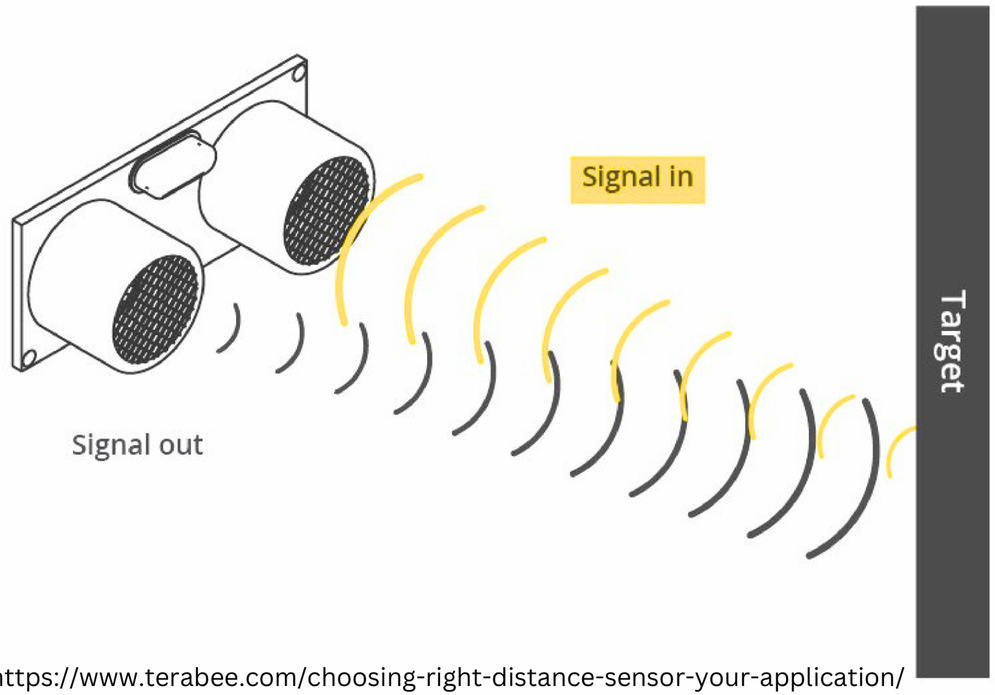
HTTPS://UK.ROBOTSHOP.COM/PRODUCTS/37-MODULES-SENSOR-KIT-ARDUINO-V2

ULTRASONIC DISTANCE SENSOR

If, instead of movement, we wanted to sense how far something was from our robot we might use an ultrasonic distance sensor.

This sensor sends out a burst of sound that is above human hearing and listens for the signal's return. The length of time between sending out the signal and hearing it lets it know just how far away an object in front of it is.

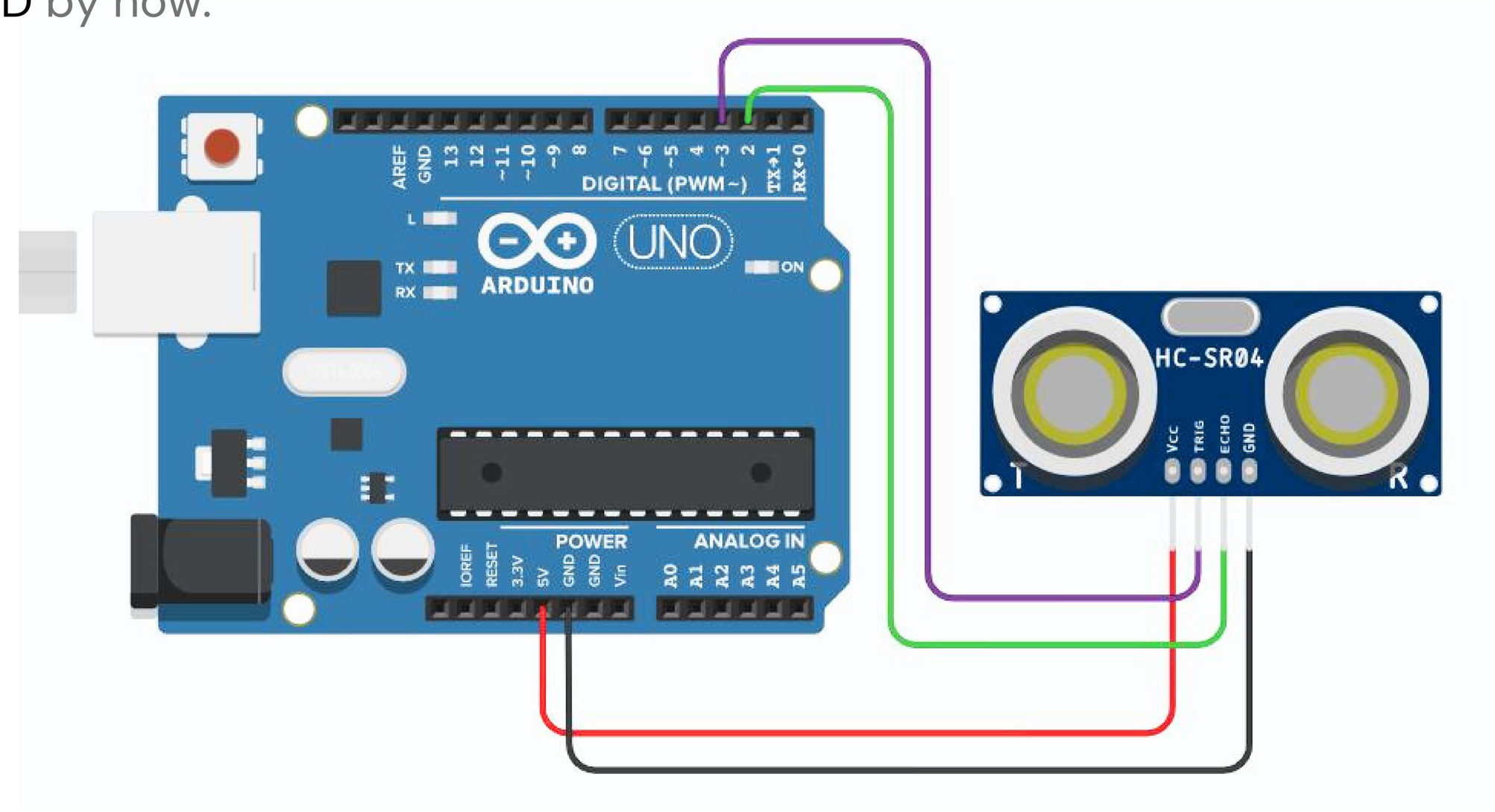
Plus it looks like eyes, and that's cute.



https://www.terabee.com/choosing-right-distance-sensor-your-application/

HOW DO WE WIRE IT UP?

Trig and Echo pins are digital, and you should be able to decipher Vcc and GND by now.



CODE FOR THE HC-SR04

Before you copy this code, lets take a look at it.

We can use #define to declare our pin variables, because we are never going to change their numbers while the program is running. This is a more efficient way to code.

The reset of the code sets the trigger pin low, sets it high for 10ms, and then sets it low again. Simple, but it looks like a mess...

sketch_mar12a.ino

```
1  //We can use #define to set our pin variables,
2  //because we are never going to change them
3  #define trigPin 4
4  #define echoPin 5
5
6  void setup() {
7    Serial.begin(115200);
8    //Trig sends the signal
9    pinMode(trigPin, OUTPUT);
10   //Echo recieves the reflected signal
11   pinMode(echoPin, INPUT);
12 }
13
14 void loop() {
15   //Long is a type of variable that holds really big numbers
16   long duration, distance;
17   //Make sure the Trig pin is off
18   digitalWrite(trigPin, LOW);  // Added this line
19   delayMicroseconds(2); // Added this line
20   //Send the signal
21   digitalWrite(trigPin, HIGH);
22   delayMicroseconds(10); // Added this line
23   //Turn off the signal
24   digitalWrite(trigPin, LOW);
25   //Read the reflected signal
26   duration = pulseIn(echoPin, HIGH);
27   //Convert it to CM
28   distance = (duration/2) / 29.1;
29   Serial.print("Distance: ");
30   Serial.println(distance);
31 }
```

LET'S TALK ABOUT PROGRAMMING BEFORE WE START

FUNCTIONS

So far we have written a lot of code to read sensors and make servos move, but it all gets crowded and it hard to tell apart.

Is there a way we could make it easier to read and easier to reuse later?

void setup() and void loop() are called Functions, and we can write our own.

sketch_mar12a.ino

```
1  //We can use #define to set our pin variables,
2  //because we are never going to change them
3  #define trigPin 4
4  #define echoPin 5
5
6  void setup() {
7    Serial.begin(115200);
8    //Trig sends the signal
9    pinMode(trigPin, OUTPUT);
10   //Echo recieves the reflected signal
11   pinMode(echoPin, INPUT);
12 }
13
14 void loop() {
15   //Long is a type of variable that holds really big numbers
16   long duration, distance;
17   //Make sure the Trig pin is off
18   digitalWrite(trigPin, LOW);  // Added this line
19   delayMicroseconds(2); // Added this line
20   //Send the signal
21   digitalWrite(trigPin, HIGH);
22   delayMicroseconds(10); // Added this line
23   //Turn off the signal
24   digitalWrite(trigPin, LOW);
25   //Read the reflected signal
26   duration = pulseIn(echoPin, HIGH);
27   //Convert it to CM
28   distance = (duration/2) / 29.1;
29   Serial.print("Distance: ");
30   Serial.println(distance);
31 }
```

MY FIRST FUNCTION

Before we setup our USD sensor, lets create a new function to put all our code in. We are also going to learn a new variable type:

long measureDistance() {}

Wait, why is this not void distance(){} ??

01-USDCode.ino

```
1  //We can use #define to set our pin variables,
2  //because we are never going to change them
3  #define trigPin 4
4  #define echoPin 5
5
6  void setup() {
7    Serial.begin(115200);
8    //Trig sends the signal
9    pinMode(trigPin, OUTPUT);
10   //Echo recieves the reflected signal
11   pinMode(echoPin, INPUT);
12 }
13
14 void loop() {
15   long distance = measureDistance();
16   Serial.print("Distance: ");
17   Serial.println(distance);
18 }
19
20 long measureDistance()
21   {//Long is a type of variable that holds really big numbers
22   long duration, result;
23   //Make sure the Trig pin is off
24   digitalWrite(trigPin, LOW);  // Added this line
25   delayMicroseconds(2); // Added this line
26   //Send the signal
27   digitalWrite(trigPin, HIGH);
28   delayMicroseconds(10); // Added this line
29   //Turn off the signal
30   digitalWrite(trigPin, LOW);
31   //Read the reflected signal
32   duration = pulseIn(echoPin, HIGH);
33   //Convert it to CM
34   result = (duration/2) / 29.1;
35   return result;
36 }
37
```

MY FIRST FUNCTION

Functions can return information to us, and the first word we use to define our function tells us what information they're going to return.

void returns nothing at all
long returns a really long number

We could also make functions using the other variable types we've learned so far: bool, int and float and they would return a number that matches their type.

01-USDCode.ino

```
1  //We can use #define to set our pin variables,
2  //because we are never going to change them
3  #define trigPin 4
4  #define echoPin 5
5
6  void setup() {
7    Serial.begin(115200);
8    //Trig sends the signal
9    pinMode(trigPin, OUTPUT);
10   //Echo recieves the reflected signal
11   pinMode(echoPin, INPUT);
12 }
13
14 void loop() {
15   long distance = measureDistance();
16   Serial.print("Distance: ");
17   Serial.println(distance);
18 }
19
20 long measureDistance()
21   {//Long is a type of variable that holds really big numbers
22   long duration, result;
23   //Make sure the Trig pin is off
24   digitalWrite(trigPin, LOW);  // Added this line
25   delayMicroseconds(2); // Added this line
26   //Send the signal
27   digitalWrite(trigPin, HIGH);
28   delayMicroseconds(10); // Added this line
29   //Turn off the signal
30   digitalWrite(trigPin, LOW);
31   //Read the reflected signal
32   duration = pulseIn(echoPin, HIGH);
33   //Convert it to CM
34   result = (duration/2) / 29.1;
35   return result;
36 }
37
```

MY FIRST FUNCTION

To make sure our long function works correctly we need to make sure it ends with an instruction to return a long value.

return result; achieves this here.

Back in void loop() we need to declare a variable that will we will use to save the returned result of our measurement .

long distance = measureDistance(); is our way of saying “We want to save a number into a variable called “distance“ and we want it to be the result of our function.

01-USDCode.ino

```
1  //We can use #define to set our pin variables,
2  //because we are never going to change them
3  #define trigPin 4
4  #define echoPin 5
5
6  void setup() {
7    Serial.begin(115200);
8    //Trig sends the signal
9    pinMode(trigPin, OUTPUT);
10   //Echo recieves the reflected signal
11   pinMode(echoPin, INPUT);
12 }
13
14 void loop() {
15   long distance = measureDistance();
16   Serial.print("Distance: ");
17   Serial.println(distance);
18 }
19
20 long measureDistance()
21   {//Long is a type of variable that holds really big numbers
22   long duration, result;
23   //Make sure the Trig pin is off
24   digitalWrite(trigPin, LOW);  // Added this line
25   delayMicroseconds(2); // Added this line
26   //Send the signal
27   digitalWrite(trigPin, HIGH);
28   delayMicroseconds(10); // Added this line
29   //Turn off the signal
30   digitalWrite(trigPin, LOW);
31   //Read the reflected signal
32   duration = pulseIn(echoPin, HIGH);
33   //Convert it to CM
34   result = (duration/2) / 29.1;
35   return result;
36 }
37
```

HOW DO I REUSE THIS?

What use is our function?

It makes our code tidier. When we look at void loop() we can get a clear idea in a more human language of what it's doing.

It makes code easier to reuse. Next time we use a distance sensor we can copy and paste our measureDistance() function [and pin definitions] into the new code and know exactly how it works, and what result it will return.

How efficient!

01-USDCode.ino

```
1   //We can use #define to set our pin variables,
2   //because we are never going to change them
3   #define trigPin 4
4   #define echoPin 5
5
6   void setup() {
7     Serial.begin(115200);
8     //Trig sends the signal
9     pinMode(trigPin, OUTPUT);
10    //Echo recieves the reflected signal
11    pinMode(echoPin, INPUT);
12  }
13
14  void loop() {
15    long distance = measureDistance();
16    Serial.print("Distance: ");
17    Serial.println(distance);
18  }
19
20  long measureDistance()
21    {//Long is a type of variable that holds really big numbers
22    long duration, result;
23    //Make sure the Trig pin is off
24    digitalWrite(trigPin, LOW);  // Added this line
25    delayMicroseconds(2); // Added this line
26    //Send the signal
27    digitalWrite(trigPin, HIGH);
28    delayMicroseconds(10); // Added this line
29    //Turn off the signal
30    digitalWrite(trigPin, LOW);
31    //Read the reflected signal
32    duration = pulseIn(echoPin, HIGH);
33    //Convert it to CM
34    result = (duration/2) / 29.1;
35    return result;
36  }
37
```

OKAY, THE VALUES FROM THIS SENSOR JUMP EVERYWHERE

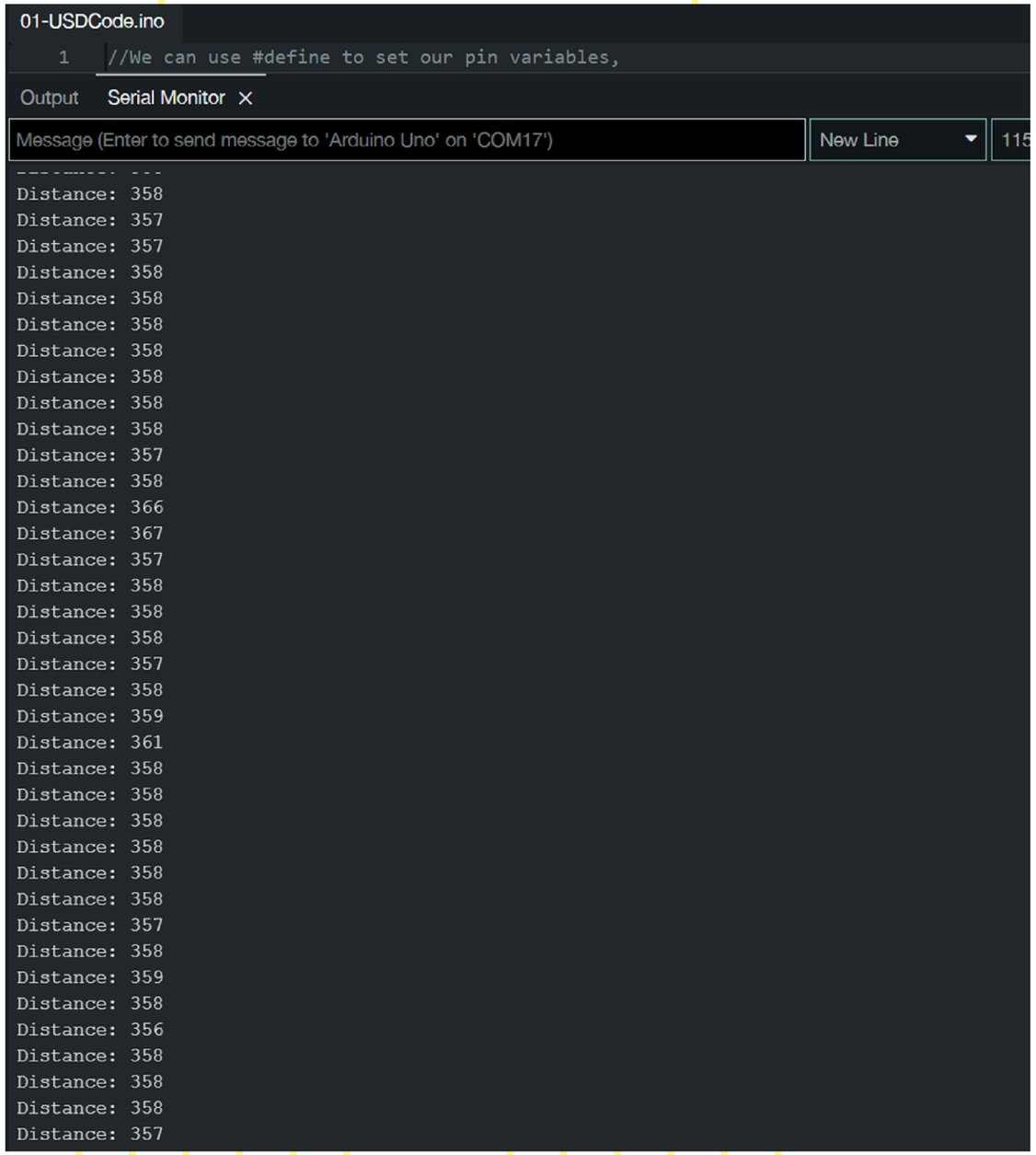
DATA SMOOTHING

In the example to the right values jump between 357 and 367. That's a lot of variation!

If we wanted to use it with map() or an if() statement we run the risk of triggering inappropriate responses with it.

What can we do?

Sometimes we might "smooth" data from an input device. To do this we need to keep track of the data we're getting in and the data that came in before it.



DATA SMOOTHING

We need a new global variable that stores our last result. Here I've added long lastDistance; for that purpose.

We create our distance reading using 75% of our last distance reading and 25% of our new distance reading

distance = (lastDistance * 0.75) + (distance * 0.25);

Writing lastDistance * 0.75 is another way of saying use 75% of lastDistance

01-USDCode.ino

```
1  //We can use #define to set our pin variables,
2  //because we are never going to change them
3  #define trigPin 4
4  #define echoPin 5
5  long lastDistance;
6
7  void setup() {
8    Serial.begin(115200);
9    //Trig sends the signal
10   pinMode(trigPin, OUTPUT);
11   //Echo recieves the reflected signal
12   pinMode(echoPin, INPUT);
13 }
14
15 void loop() {
16   long distance = measureDistance();
17   distance = (lastDistance * 0.75) + (distance * 0.25);
18   lastDistance = distance;
19   Serial.print("Distance: "); Serial.println(distance);
20 }
21
22 long measureDistance()
23   {//Long is a type of variable that holds really big numbers
24   long duration, result;
25   //Make sure the Trig pin is off
26   digitalWrite(trigPin, LOW);  // Added this line
27   delayMicroseconds(2); // Added this line
28   //Send the signal
29   digitalWrite(trigPin, HIGH);
30   delayMicroseconds(10); // Added this line
31   //Turn off the signal
32   digitalWrite(trigPin, LOW);
33   //Read the reflected signal
34   duration = pulseIn(echoPin, HIGH);
35   //Convert it to CM
36   result = (duration/2) / 29.1;
37   return result;
38 }
```

DATA SMOOTHING

Next we need to update our lastDistance variable for next time.

lastDistance = distance; takes care of that.

Now we can load this onto the Arduino and see how it changes the result. Yes, there is still some noise, but it is greatly reduced.

You can experiment with smoothing by changing the amount you * lastDistance and distance by, but it should always add up to 1.0

01-USDCode.ino

```
1  //We can use #define to set our pin variables,
2  //because we are never going to change them
3  #define trigPin 4
4  #define echoPin 5
5  long lastDistance;
6
7  void setup() {
8    Serial.begin(115200);
9    //Trig sends the signal
10   pinMode(trigPin, OUTPUT);
11   //Echo recieves the reflected signal
12   pinMode(echoPin, INPUT);
13 }
14
15 void loop() {
16   long distance = measureDistance();
17   distance = (lastDistance * 0.75) + (distance * 0.25);
18   lastDistance = distance;
19   Serial.print("Distance: "); Serial.println(distance);
20 }
21
22 long measureDistance()
23   {//Long is a type of variable that holds really big numbers
24   long duration, result;
25   //Make sure the Trig pin is off
26   digitalWrite(trigPin, LOW);  // Added this line
27   delayMicroseconds(2); // Added this line
28   //Send the signal
29   digitalWrite(trigPin, HIGH);
30   delayMicroseconds(10); // Added this line
31   //Turn off the signal
32   digitalWrite(trigPin, LOW);
33   //Read the reflected signal
34   duration = pulseIn(echoPin, HIGH);
35   //Convert it to CM
36   result = (duration/2) / 29.1;
37   return result;
38 }
```

“WHAT DO WE DO NOW WE HAVE MORE RELIABLE DATA?”

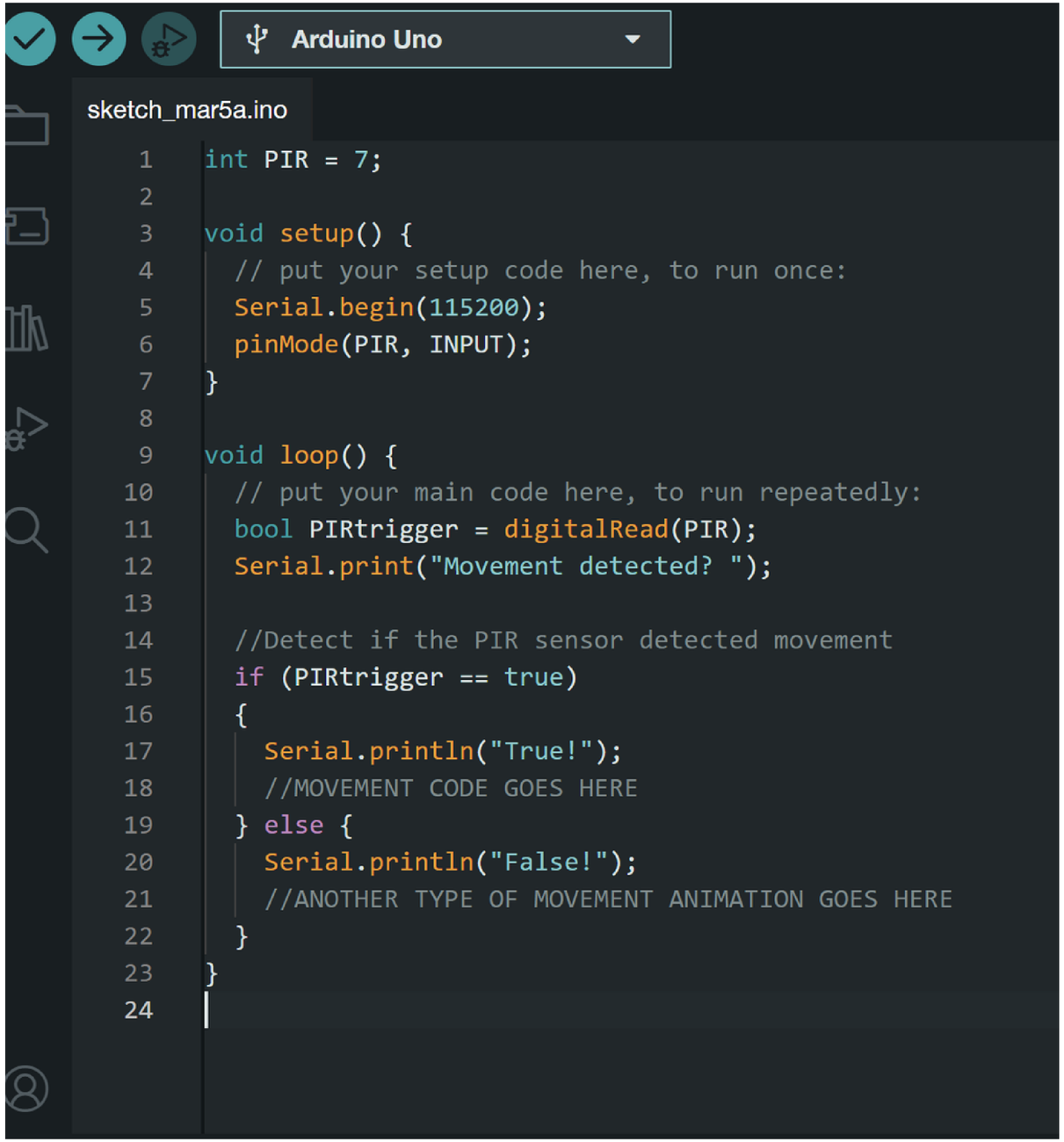
REVISITING IF{} STATEMENTS

Last week we looked at if statements. Lets revisit that now.

We learned we can use

```
if (value == true) {
    //run commands
} else {
   //run other commands
}
```

We make a choice, but we only have two options here. Can we do more?



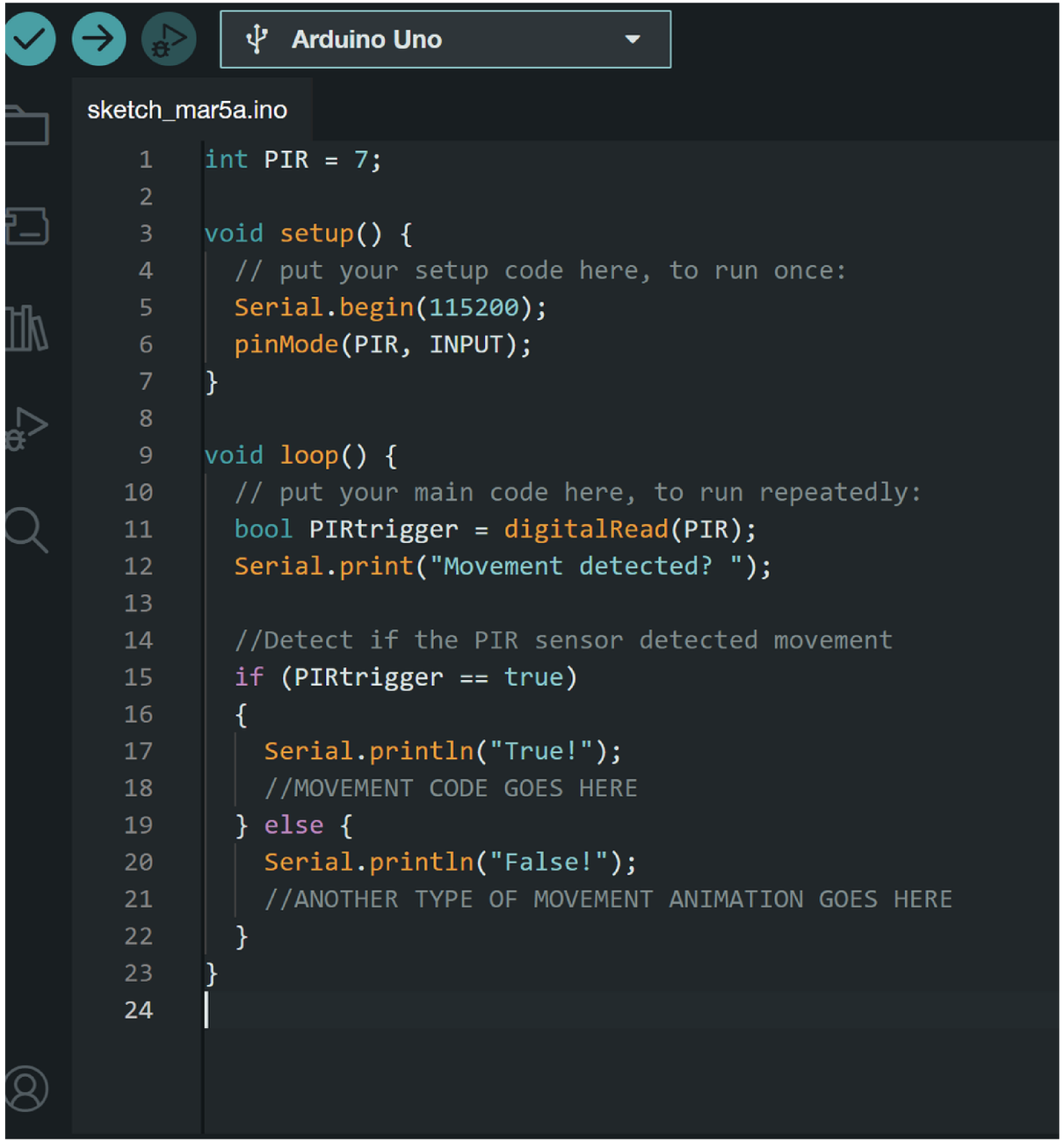
REVISITING IF{} STATEMENTS

Yes, of course we can do more than that!

We can use other mathematic symbols to test a value and decide what to do.

> [greater than]
< [less than]
!= [does not equal
>= [greater than or equal to]
<= [less than or equal to]

All these symbols are useful with if() statements



REVISITING IF{} STATEMENTS

Lets just focus on our
void loop() for now.

First we test:
if (distance > 200) {

We can use this to define our robots behaviour for all values over 200.

Next:
else if ()

Until now else has just been used as “in all other cases, do this instead” but we can add another if() statement there with else if()! If our first if statement is false, we can try again with a new test.

01-USDCode.ino

```
15  void loop() {
16    long distance = measureDistance();
17    distance = (lastDistance * 0.75) + (distance * 0.25);
18    lastDistance = distance;
19
20    if (distance > 200) {
21      // Code for if something is far away
22    } else if (distance < 200 && distance > 100) {
23      //Code for if something is a medium distance
24    } else if (distance < 100 && distance > 40) {
25      //Code if something is close
26    } else {
27      //Code if someting is REALLY close
28    }
29
30    Serial.print("Distance: "); Serial.println(distance);
31  }
32
33  long measureDistance()
34    {//Long is a type of variable that holds really big numb
35    long duration, result;
36    //Make sure the Trig pin is off
37    digitalWrite(trigPin, LOW);  // Added this line
38    delayMicroseconds(2); // Added this line
39    //Send the signal
40    digitalWrite(trigPin, HIGH);
41    delayMicroseconds(10); // Added this line
42    //Turn off the signal
```

REVISITING IF{} STATEMENTS

The whole line reads:
else if (distance > 200 && distance > 100) { }

In English we sometimes use & as a shortcut for the word “and”.

In programming using && means “test the values on our left, and the values to the right. If both are true then you can run this code.”

Any code we put between the curly brackets { } will only run if distance is both less than 200 and more than 100, so between 101 and 199.

01-USDCode.ino

```
15  void loop() {
16    long distance = measureDistance();
17    distance = (lastDistance * 0.75) + (distance * 0.25);
18    lastDistance = distance;
19
20    if (distance > 200) {
21      // Code for if something is far away
22    } else if (distance < 200 && distance > 100) {
23      //Code for if something is a medium distance
24    } else if (distance < 100 && distance > 40) {
25      //Code if something is close
26    } else {
27      //Code if someting is REALLY close
28    }
29
30    Serial.print("Distance: "); Serial.println(distance);
31  }
32
33  long measureDistance()
34    {//Long is a type of variable that holds really big numb
35    long duration, result;
36    //Make sure the Trig pin is off
37    digitalWrite(trigPin, LOW);  // Added this line
38    delayMicroseconds(2); // Added this line
39    //Send the signal
40    digitalWrite(trigPin, HIGH);
41    delayMicroseconds(10); // Added this line
42    //Turn off the signal
```

REVISITING IF{} STATEMENTS

Ordering is important here.

We want to test only in ascending or descending values. This makes it easier to read and makes sure we're not accidentally catching the wrong values.

Also, we can only use an else after an if or an else if, but not in-between an if and an else if.

We write the code as:
if then else if then else.

01-USDCode.ino

```
15  void loop() {
16    long distance = measureDistance();
17    distance = (lastDistance * 0.75) + (distance * 0.25);
18    lastDistance = distance;
19
20    if (distance > 200) {
21      // Code for if something is far away
22    } else if (distance < 200 && distance > 100) {
23      //Code for if something is a medium distance
24    } else if (distance < 100 && distance > 40) {
25      //Code if something is close
26    } else {
27      //Code if someting is REALLY close
28    }
29
30    Serial.print("Distance: "); Serial.println(distance);
31  }
32
33  long measureDistance()
34    {//Long is a type of variable that holds really big numb
35    long duration, result;
36    //Make sure the Trig pin is off
37    digitalWrite(trigPin, LOW);  // Added this line
38    delayMicroseconds(2); // Added this line
39    //Send the signal
40    digitalWrite(trigPin, HIGH);
41    delayMicroseconds(10); // Added this line
42    //Turn off the signal
```

NEXT WEEK IS WEEK 6

NO CLUB OR CLASSES NEXT WEEK

Weeks 7 - 10

PROJECT TIME

Do you have something you want to make now you've learned this much?
Do you have an assignment you want to add electronics to?
Do you want to hang out, experiment more, and get advice?
Are you lonely and just need somewhere where people are nice?

JOIN US FOR PROJECT TIME!

NEXT TERM

WE CONTINUE LEARNING!

Other things we could learn in the future:

Bluetooth / RF communication

Newer microcrontrollers [Fether, M5, etc]

Triggering high voltage [relays, transistors, MOSFETs]

LEDs and the like [DotStar / Neopixels]

Computer / Arduino communication [Ableton or Processing / Serial]

Inverse Kinematics [maths for movement - I don’t know how this works]

NEXT TERM

WE WILL FOCUS ON...

Do you want to continue to focus on making robots?
Should we look at a broader range of systems?
Do you want to look at getting data from a computer into the Arduino or vice versa?
Should we focus on a different kind of output?
Maybe we could make instruments?

We are here to make learning more interesting for you!