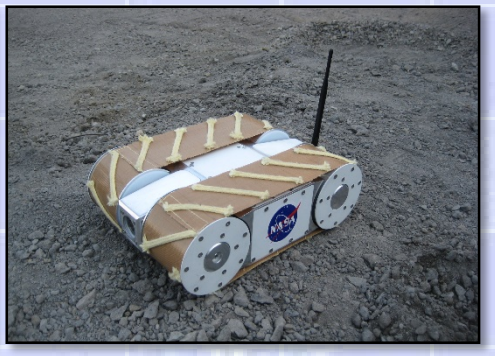





How Would You Measure the Moon?

MAKE it NASA

How Would You Measure the Moon?

Agenda

- Introductions and NASA Content Overview
- Applying Design Thinking to a Making Activity
  - Discover
  - Interpret
  - Ideate
  - Experiment - Building a Sample Payload
  - Experiment - Programming the Microcomputer
  - Evolve
- Conclusion

Share Your NASA Experience

Share your NASA experiences, pictures and videos

- Facilitators participating at NASA professional development workshops
- Students using NASA content
- Your organization connecting with NASA Subject Matter Experts

@NASAGRC_Edu – NASA Glenn Office of STEM Engagement on Twitter

@NASAGlenn official accounts: 

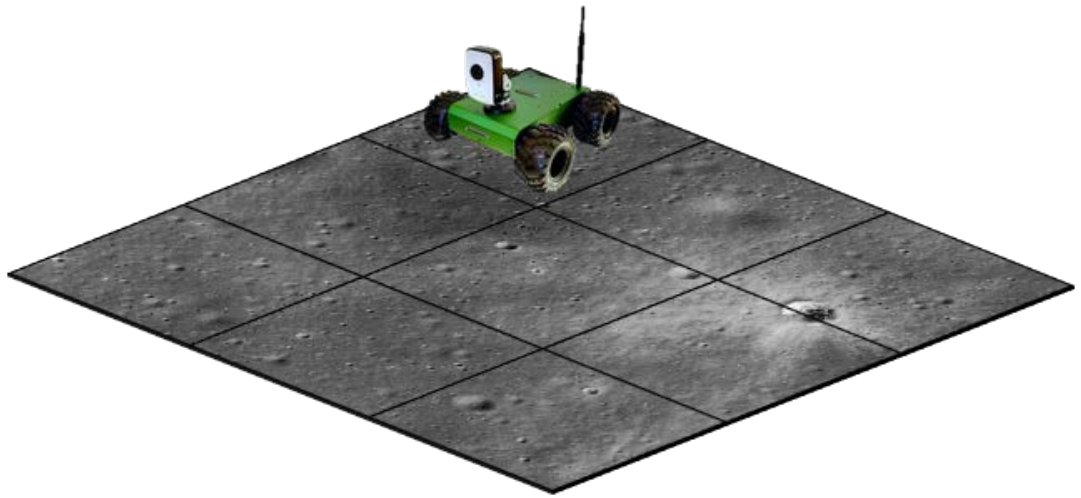
@NASAedu – NASA Office of STEM Engagement official Twitter account

- Be sure to use the hashtag #NASAGlennSTEM

How Would You Measure the Moon?

- Student teams plan and conduct a scientific mission on the lunar surface.

- They design a scientific payload, mount it on a model lunar rover, and conduct the mission, transmitting at least 9 data points back to the team.



The Student Journal

- This document will help students organize their thoughts and track their progress through the challenge.

- Instructions on the left-side pages; student work on the right-side pages.

Working in Teams

Making projects are all about individual creativity, however the best ideas rarely come from just one person. For this project, it is highly encouraged to have students work in teams of two or more.

Assign roles to team members or have students select their own roles.

- **Design Engineer** – sketches, outlines, patterns, or plans the ideas the team generates
- **Technical Engineer** – assembles, maintains, repairs, and modifies the structural components of the design
- **Operations Engineer** – sets up and operates the prototype to determine what parts work like they should and what can be improved.
- **Technical Writer/Videographer** – records and organizes information, data, and prepares documentation, via pictures and/or video to be reported and published.

Design Your Mission Patch

- Establish a mission/project name – Many NASA missions are named based on the work they do.

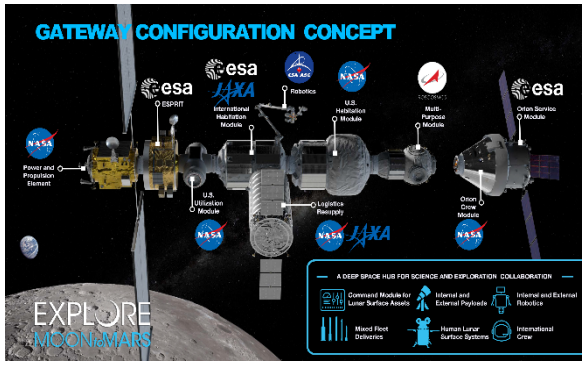
- Design a mission patch – Scientists and engineers that work on NASA missions and spacecraft are unified under mission patches that are designed with symbols and artwork to identify the group's mission.

- Create a vision statement – This is a short inspirational sentence or phrase that describes the core goal of the team's work. NASA's current vision statement is:
  - "We reach for new heights and reveal the unknown for the benefit of humankind."

NASA's Past, Present and Future at the Moon

- 2019 marks the 50th anniversary of the Apollo 11 moon landing.
- 6 Apollo missions landed on the moon between 1969 and 1972.
- NASA is supporting special events over the next 4 years to commemorate the 50th anniversaries of all of the Apollo missions.

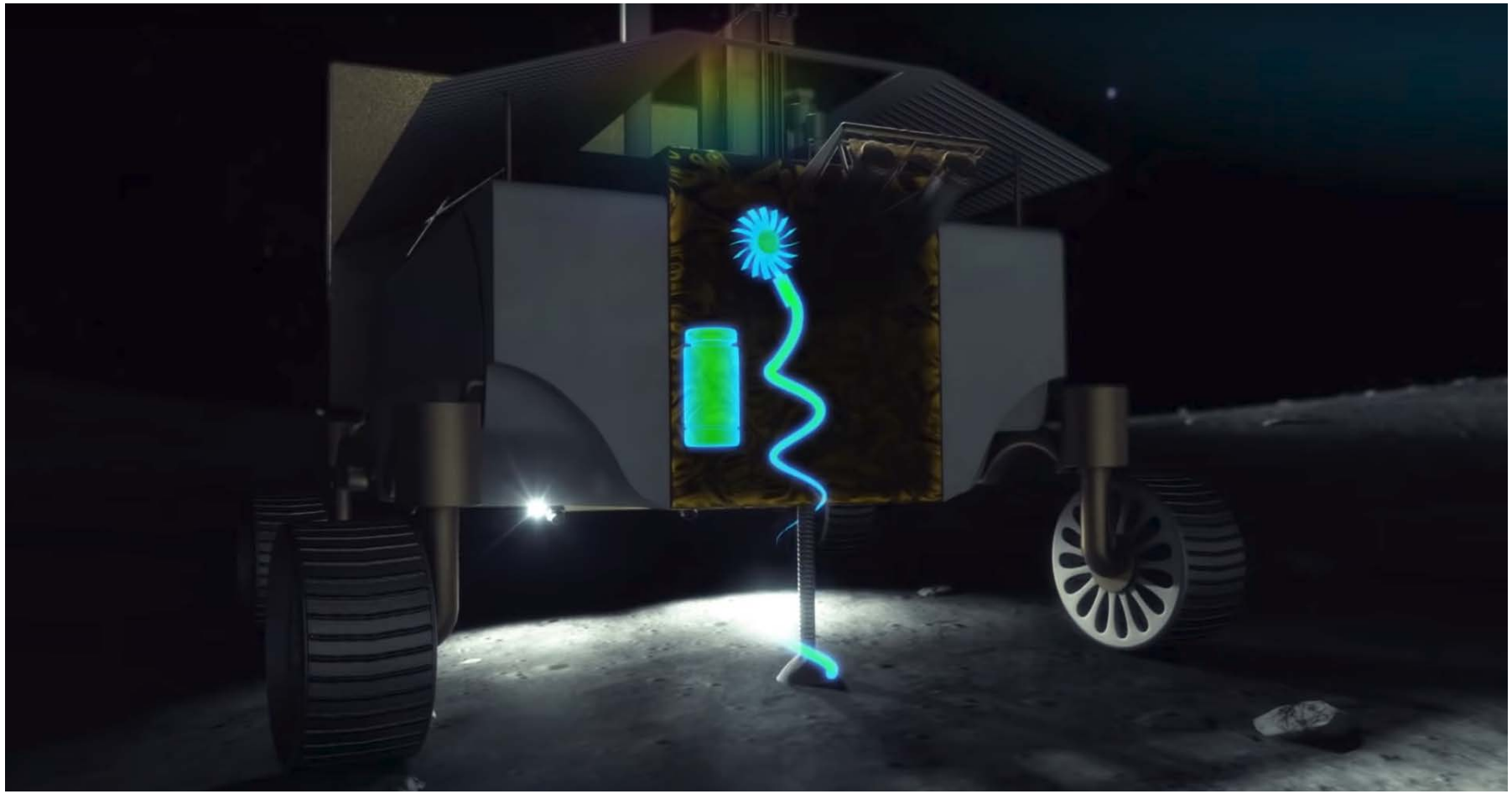
- On February 21, 2019, NASA announced a selection of 12 NASA science and technology demonstration payloads to fly to the Moon on commercial landers.
- NASA Glenn is developing one of the payloads, a solar cell demonstration platform that will test advanced solar arrays for longer mission duration.
- On March 26, 2019, the Vice President called for Astronauts to return to the Moon by 2024.



- NASA is developing plans for a lunar-orbiting outpost called the Gateway.
- This platform will consist of a power and propulsion element, habitation, communications, logistics and airlock capabilities. It will allow robotic (and potentially human) missions to be directly deployed to the surface.
- Gateway will help us explore the Moon and its resources and ultimately translate that experience toward human missions to Mars.

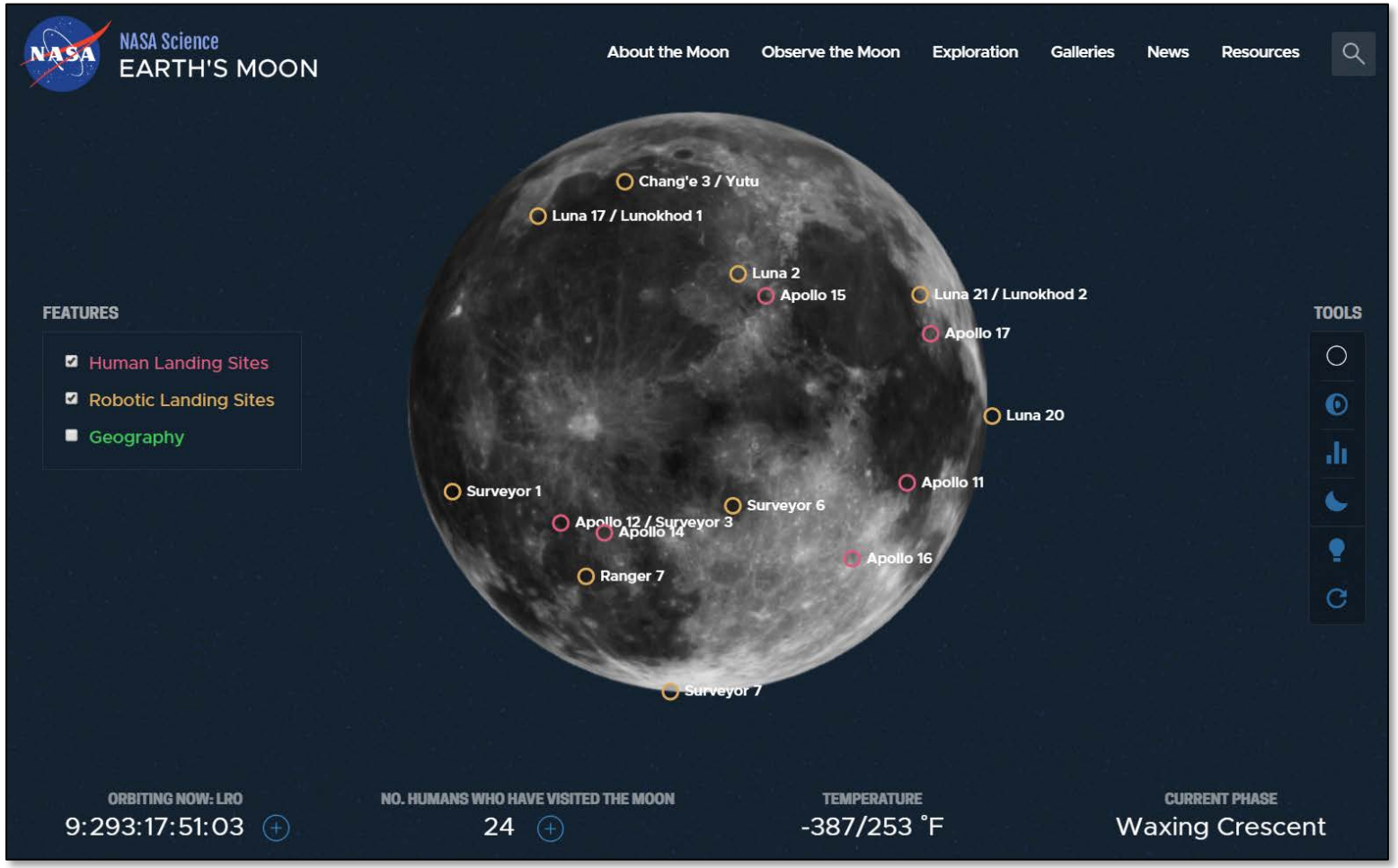
Why Send Robots to Explore the Moon?

https://youtu.be/wJ0ia4M2dxs



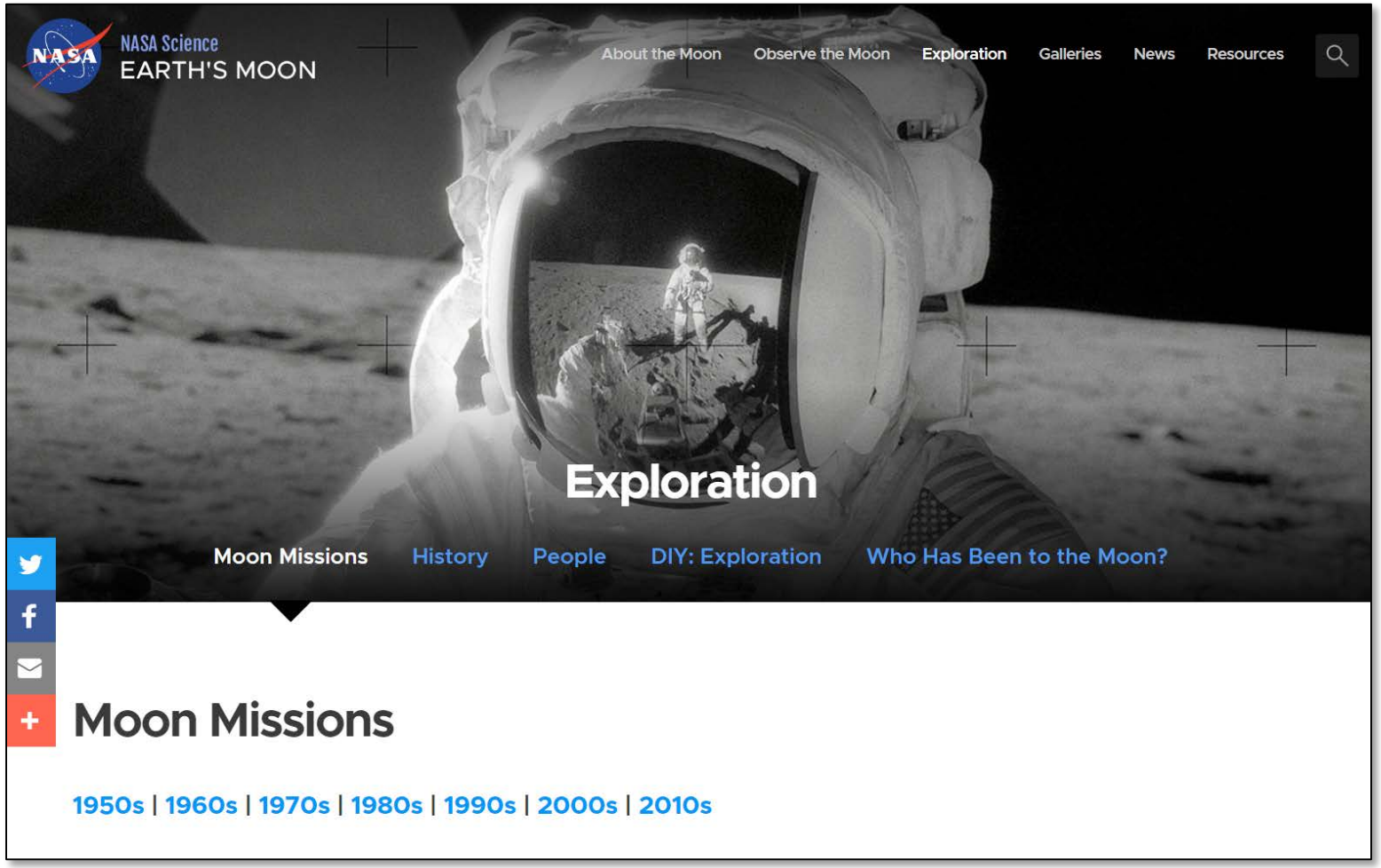
The Basics of the Moon

https://moon.nasa.gov/



Missions to the Moon

https://moon.nasa.gov/exploration/moon-missions/



The Design Thinking Process

- This process can be used to solve any problem that requires designing and making a solution.

- Adapted for student use from *Design Thinking for Educators*

https://designthinkingforeducators.com/design-thinking/.

- To make a great solution, first you have to understand the problem.

- Make a list of challenges that need to be solved. These are questions that need answers. (How can…? How would…? How does…?)

- Next, brainstorm any criteria, constraints, and barriers that are part of this problem.
  - **Criteria** are things your solution has to be able to do.
  - **Constraints** are things your solution must not do.
  - **Barriers** are things that could prevent you from finishing your solution.

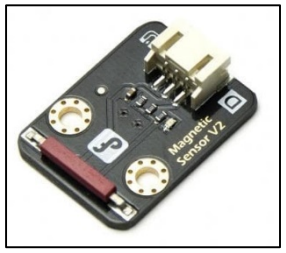
- Students will attach their payload to a rover. The rover can be built, 3D printed, or purchased (any remote controlled car could work).

- They will have to determine how to control their rover and deploy their payload to the surface.

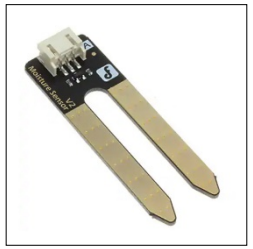
- Students will build a sensor-microcontroller-transmission system using COTS parts.

Ambient Light Sensor

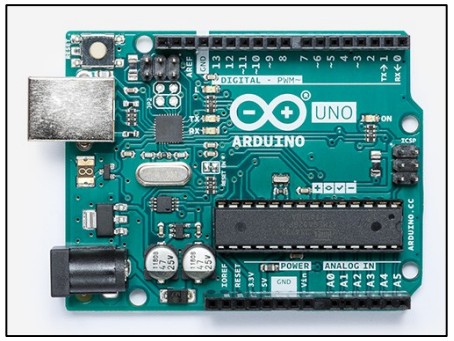
Temperature Sensor



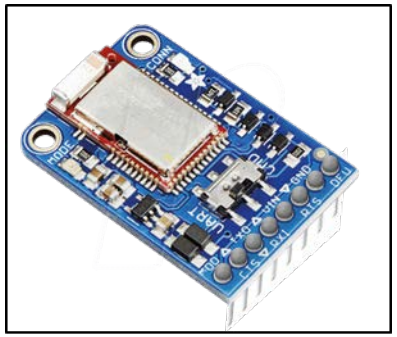
Magnetism Sensor



Soil Moisture Sensor



Microcontroller



Bluetooth Radio

Some examples of challenge questions:

- Type of Measurement
  - What is the surface temperature?
  - What is the ambient temperature?
  - Is there any seismic (vibration) activity on the surface?
  - Is there a magnetic field?
  - How much light is available on the surface?
  - How reflective is the surface?

- Method of Deployment
  - Can we take our data from on-board the rover?
  - Do we need to deploy our sensor on an arm?
  - How will we get data from multiple locations on the surface?

- Transmission Format
  - Can we transmit to a device using radio signal?
  - Should we use a light-based signal instead?

Interpret

• **I learned something.** *How do I interpret it?*

- To begin solving this challenge, you will need to find answers to the criteria, constraints and barriers you listed.

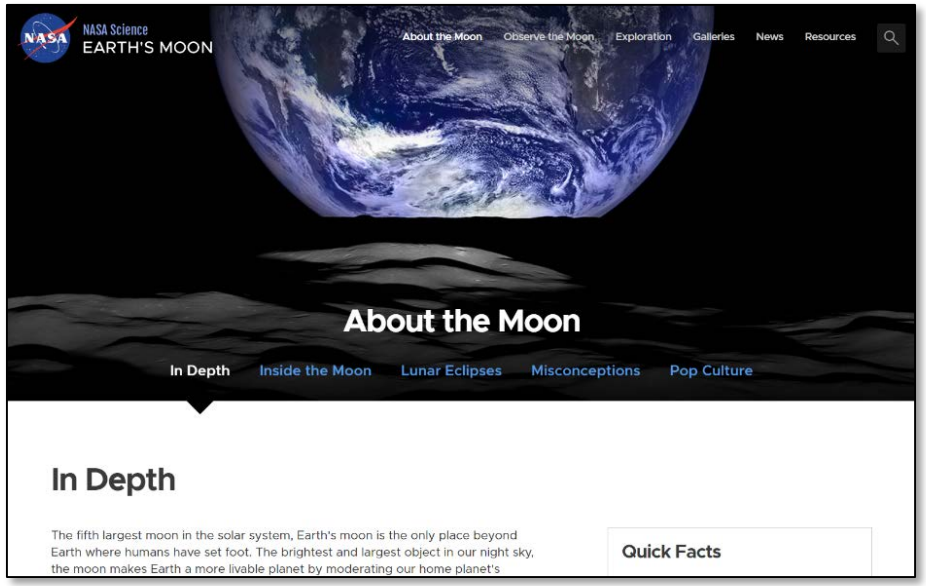
- Brainstorm resources you could use to get more facts about your challenge. You could use books, experts or trustworthy sources on the Internet.

- Access the sources you listed. Write down the key ideas that you learned. Think about how those ideas should affect your solution.

NASA Resources for This Activity

Sample of NASA Resources

- https://www.nasa.gov/moon
- https://solarsystem.nasa.gov/moons/earths-moon/
- https://moon.nasa.gov/resources/
- https://www.nasa.gov/topics/moon-to-mars
- https://www.nasa.gov/topics/moon-to-mars/videos
- https://www.nasa.gov/education/materials/
- https://www.jpl.nasa.gov/edu/teach/tag/search/Earth%27s+Moon



NASA Resources for This Activity

- A sheet of NASA website resources has been provided for you.
- Review these resources and determine which will be most helpful for your students. Things to consider:
  - What problems are your students trying to solve? Do the sources help answer their questions?
  - Are they at an appropriate reading level for your students? Will you have to help them through?
- NASA has a lot of information, but also consider additional resources from other reputable sources

Ideate

• **I see an opportunity.** *What do I create?*

- Use the things you learned to create a sketch of your solution. Label all major parts.

- Describe how it works.

- Build your solution for the first time. This model is called a **prototype**.

- Think about what materials you need to make each part of your model.

- Take a picture of your model. Add it to your journal.

Resources can come from a variety of places.

Six Suggested Starter Categories for an Elementary Makerspace (Fontichiaro, 2016)

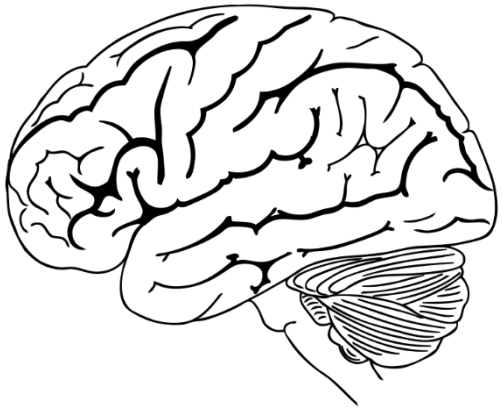
Craft	Engineering	Code	Circuits (& Computing)	Digital Design	Needle & Thread
• Origami • Modeling Clay • Wikki Stix • Scrapbooking • Junk Box creations • Recycled Materials • Challenges	• Tinkertoys • LEGO • K'Nex • BuildWithChrome.com	Robots: • Dash & Dot • Sphero • Ozobot Animation: • Scratch • Blockly • Hour of Code Apps: • Hopscotch • Scratch Jr. • Daisy the Dinosaur	• Arduino • Raspberry Pi • Lego Mindstorms • Snap Circuits • Squishy Circuits • littleBits • K'Nex with electrical components • Circuit blocks	• Canva.com • Picmonkey.com • Makebeliefcomix.com • Pixton.com • 3-D printers • Laser cutters	• Hand sewing • Machine sewing • Knitting • Crochet • Fashion Hacking • Embroidery • Cross Stitch

Building a Sensing System

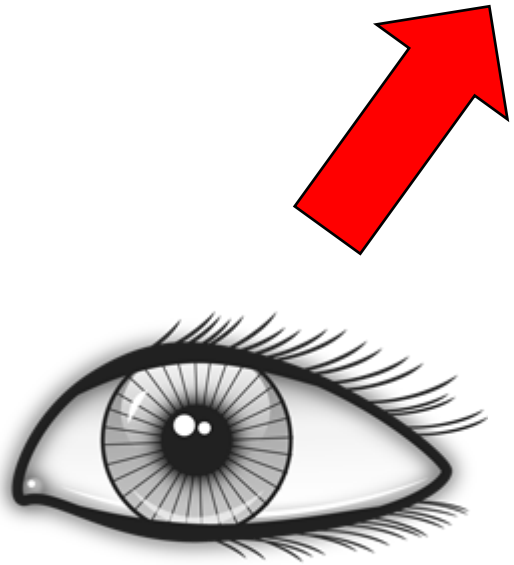
- For the purposes of this project, we will build and program a sensor array from a starter kit.
- Students are encouraged to:
  - Expand the build - add more controls or indicators
  - Modify it – use a different programming sequence
  - Customize it – use a different system altogether!
- Since this is a making activity, you should feel open to make changes based on what works for you.

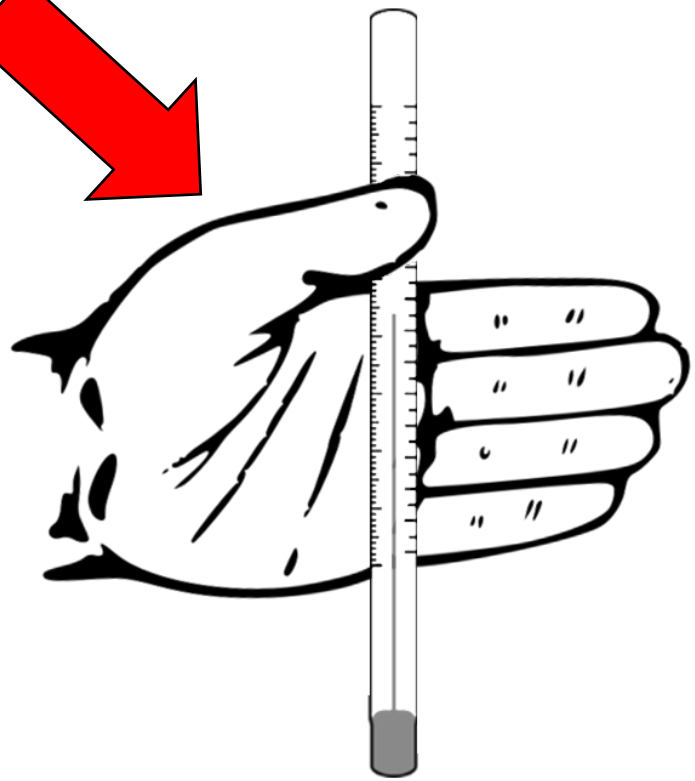
Humans and Robots - Not so Different

Step 3: Eye passes data to brain for analysis



Step 1: Brain tells hand to move thermometer into place

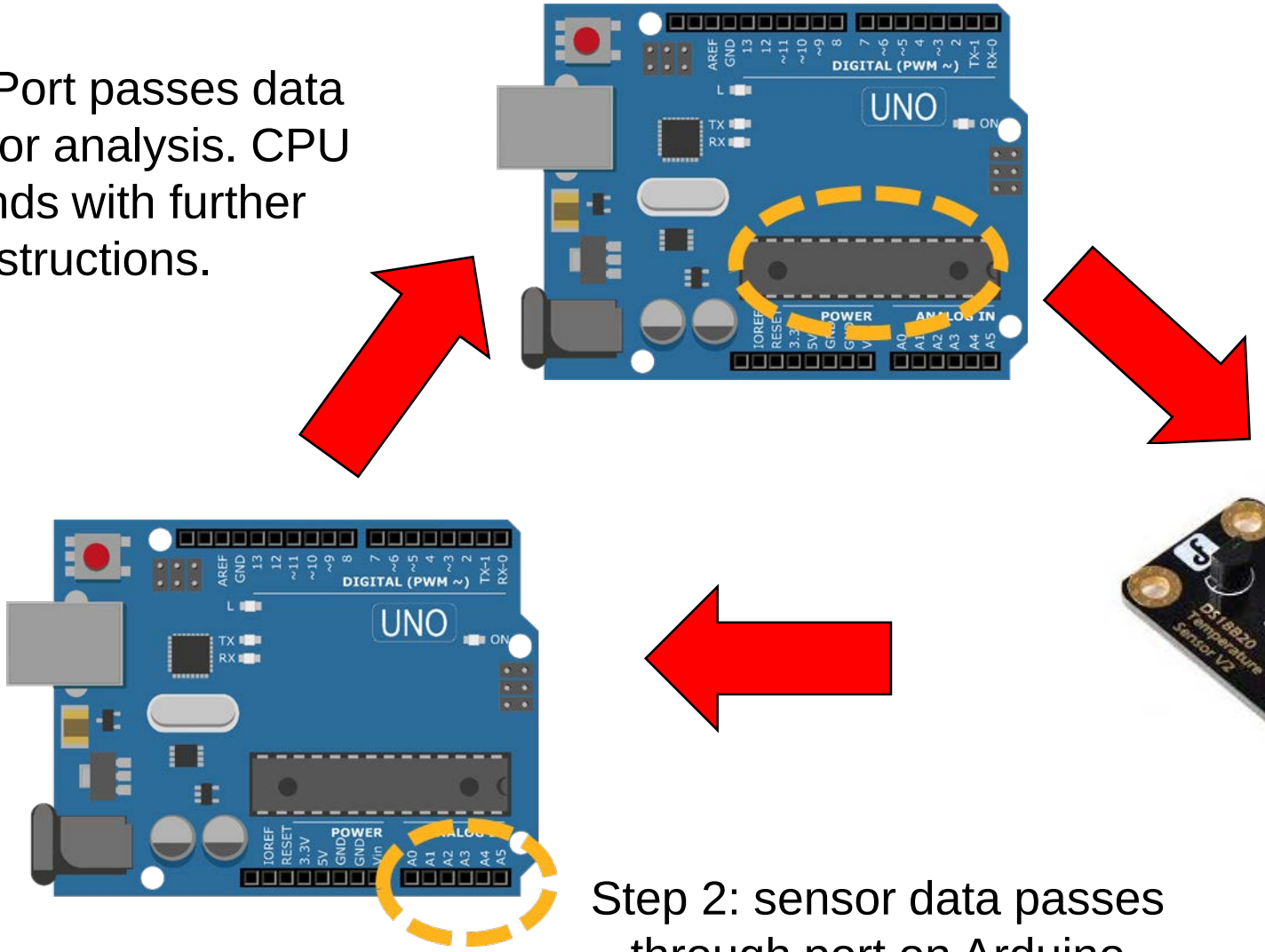




Step 2: Thermometer data passes through eye

Humans and Robots - Not so Different

Step 3: Port passes data to CPU for analysis. CPU responds with further instructions.



Step 1: Central Processing Unit (CPU) on Arduino tells servo motor to move sensor into place.

Step 2: sensor data passes through port on Arduino

Humans and Robots - Not so Different

Human

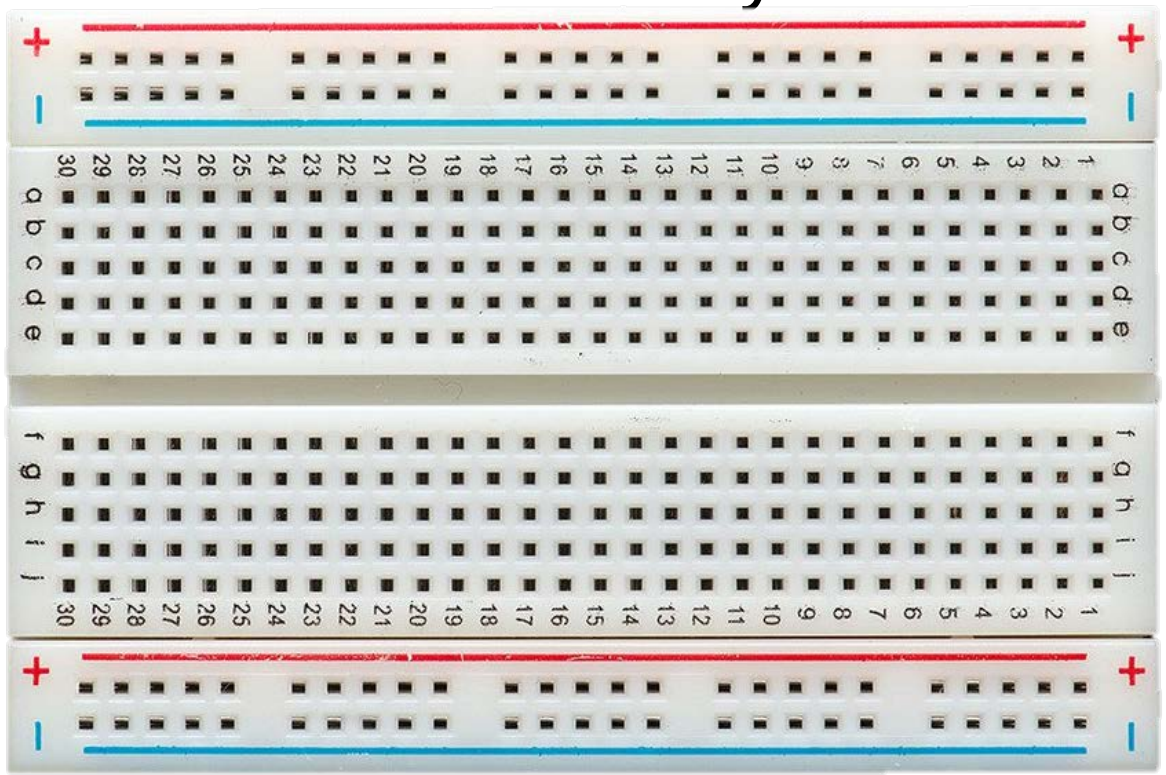
- Blood vessels:
  - Arteries take blood to cells
  - Veins return blood to source
- Nerves
  - Motor nerves send command signals from the brain to the body.
  - Sensory nerves take data from the body to the brain

Robot

- Power Wires:
  - (**Red**) Connected to 5V, give electricity from the source
  - (**Black**) Connected to GND, take electricity back to the source
- Signal Wires
  - (**Green**) Connected to Digital terminals, give start/stop commands to various parts of the robot
  - (**Blue**) Connected to Analog terminals, take ranged data from sensors

Building Circuits With Breadboards

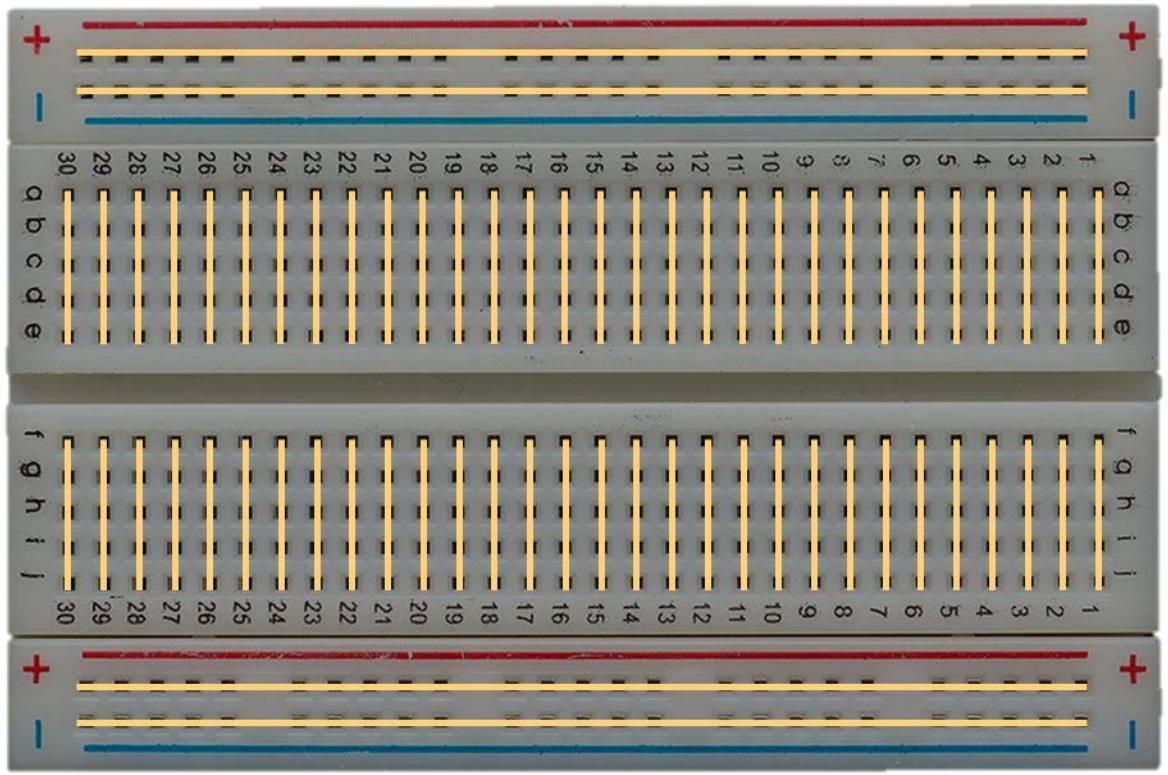
A breadboard is a convenient way to connect everything.



You can’t see them, but there are lots of wires inside it

Building Circuits With Breadboards

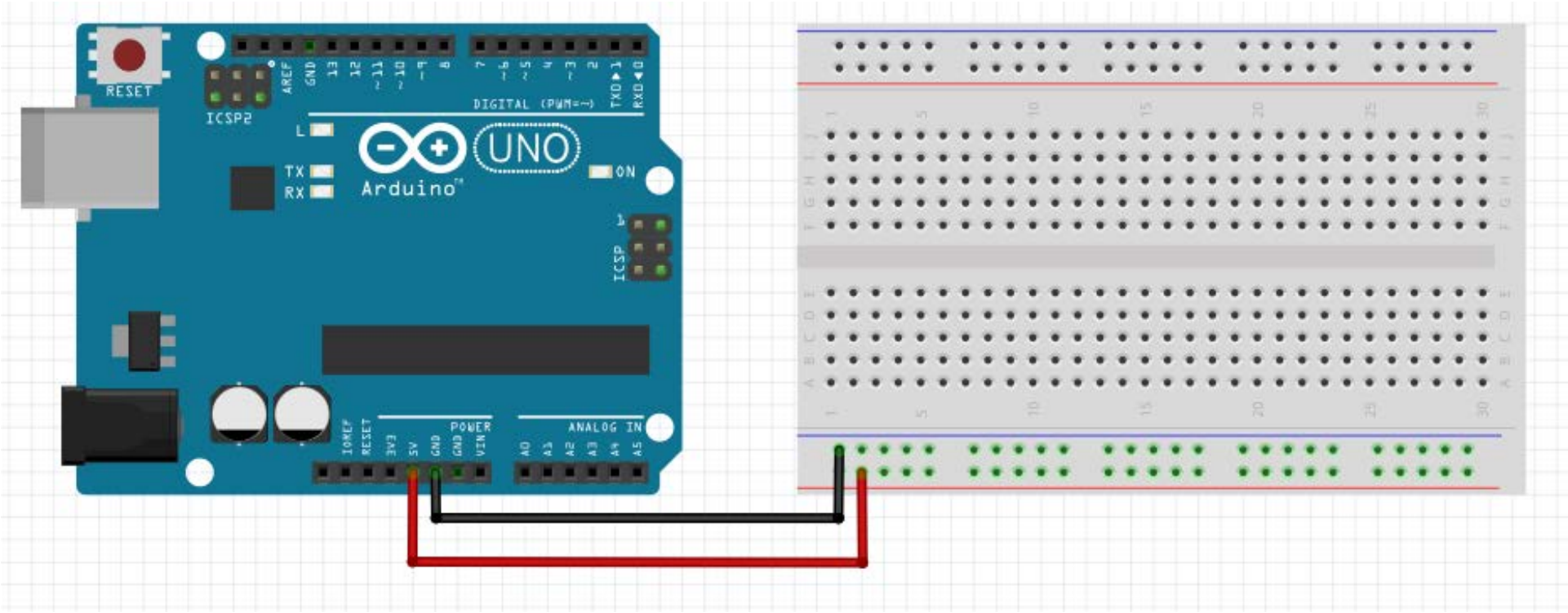
The two top and two bottom rows are connected across the board.



All of the numbered channels are connected a-e and f-j with a gap in the middle. This gap can be bridged with compatible components.

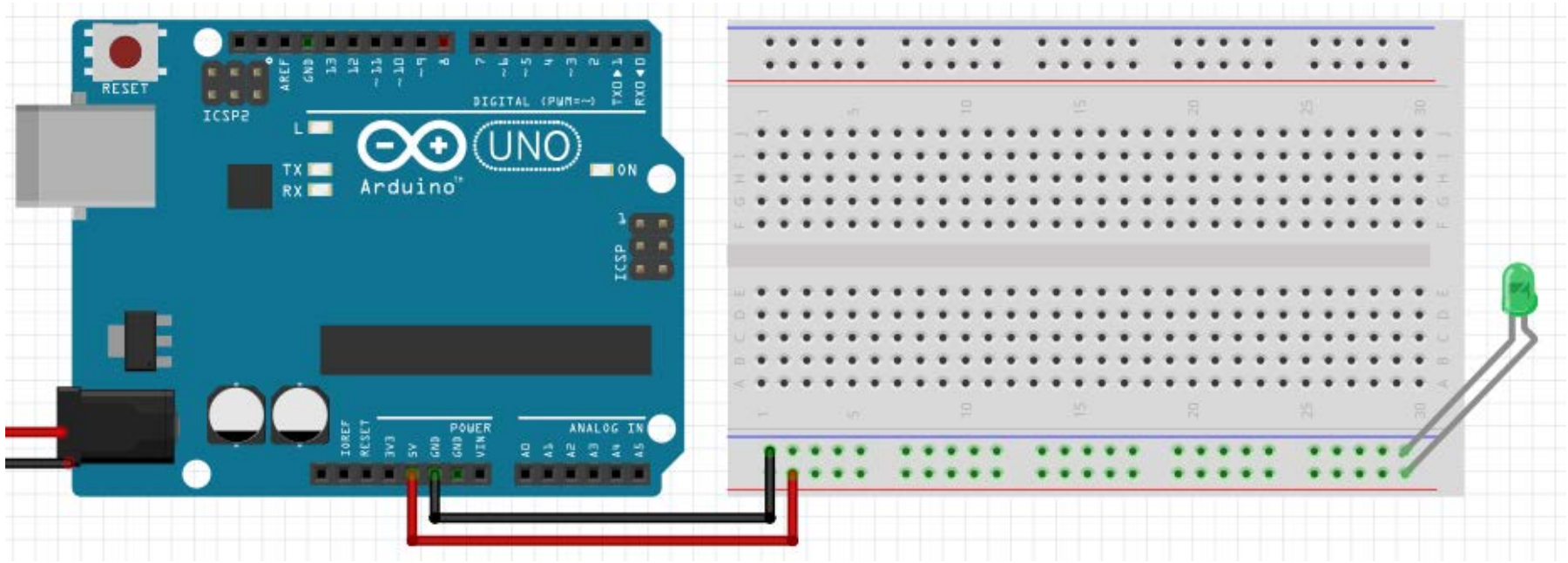
Step 1: Connect 5V and GND

These 5-volt (**5V**) and Ground (**GND**) terminals are for things that are always powered (like the motor and Bluetooth radio).



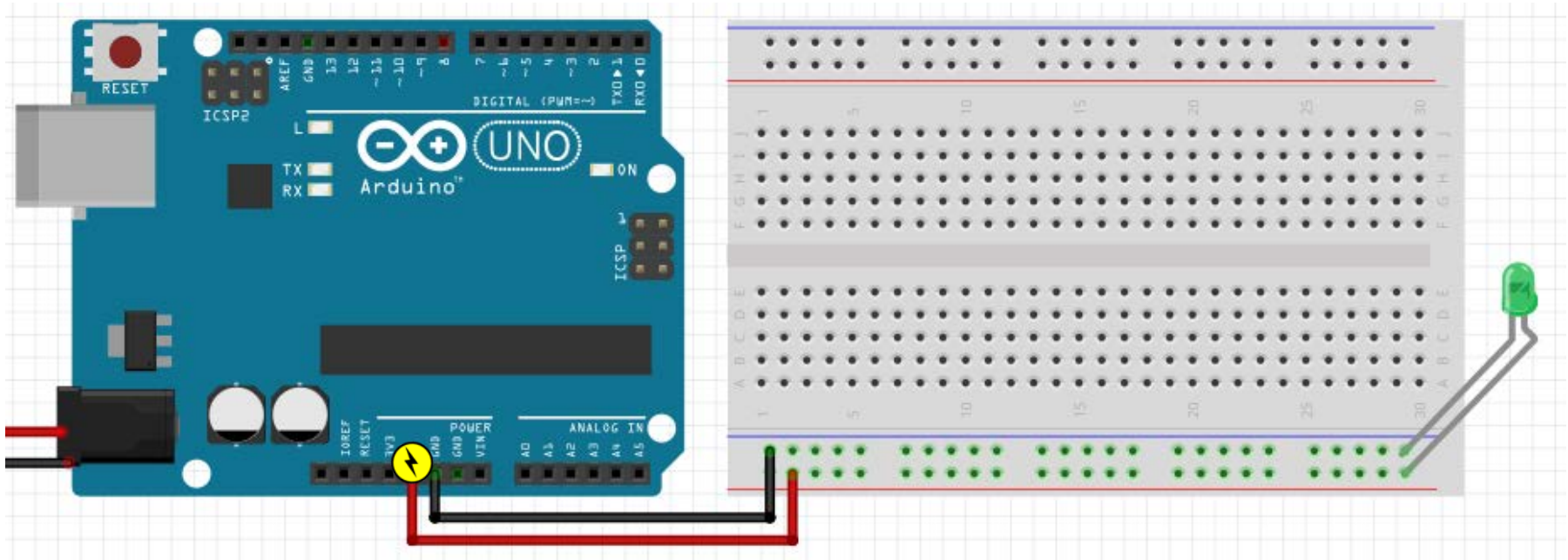
Step 2: Connect a Light Emitting Diode (LED) with Resistor

An LED is an easy way to test if your circuit is working. These LEDs come with a built-in resistor to prevent it from overheating.



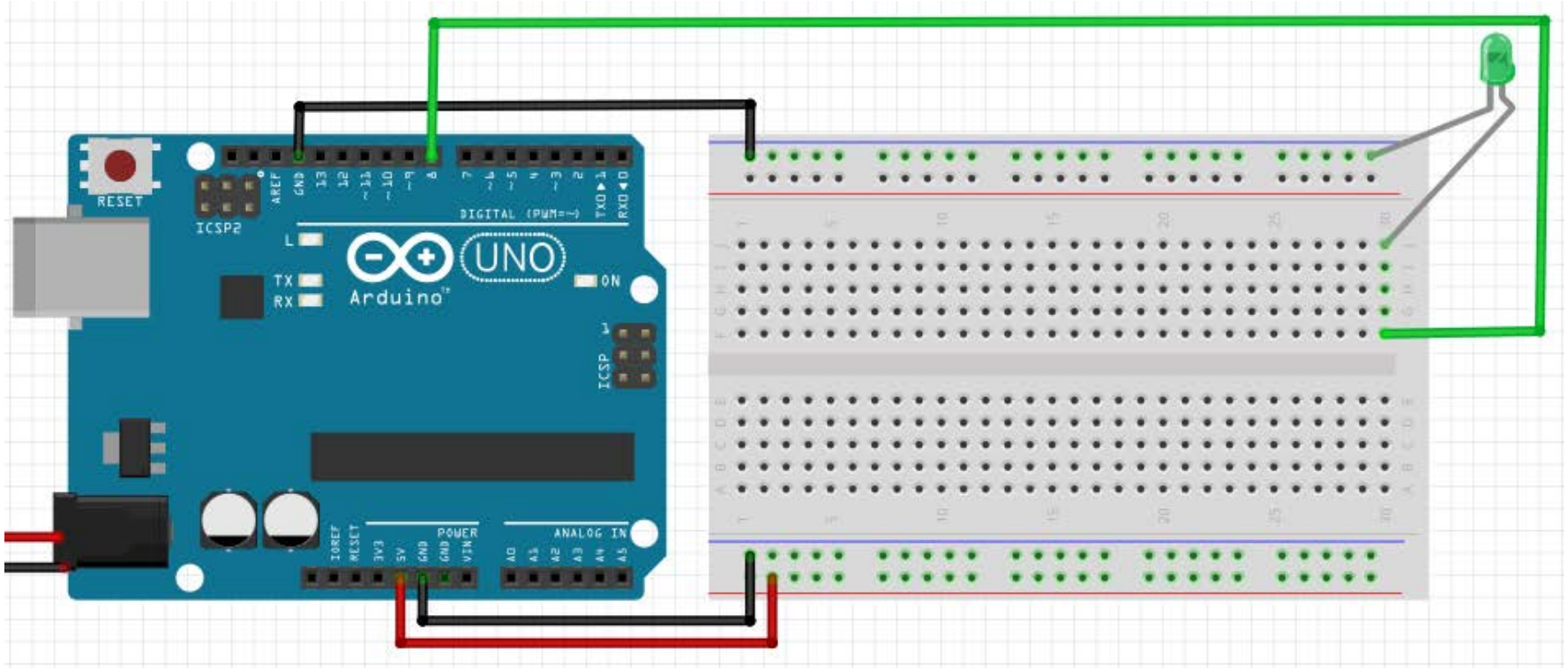
Step 2: Connect a Light Emitting Diode (LED) with Resistor

An LED is an easy way to test if your circuit is working. These LEDs come with a built-in resistor to prevent it from overheating.



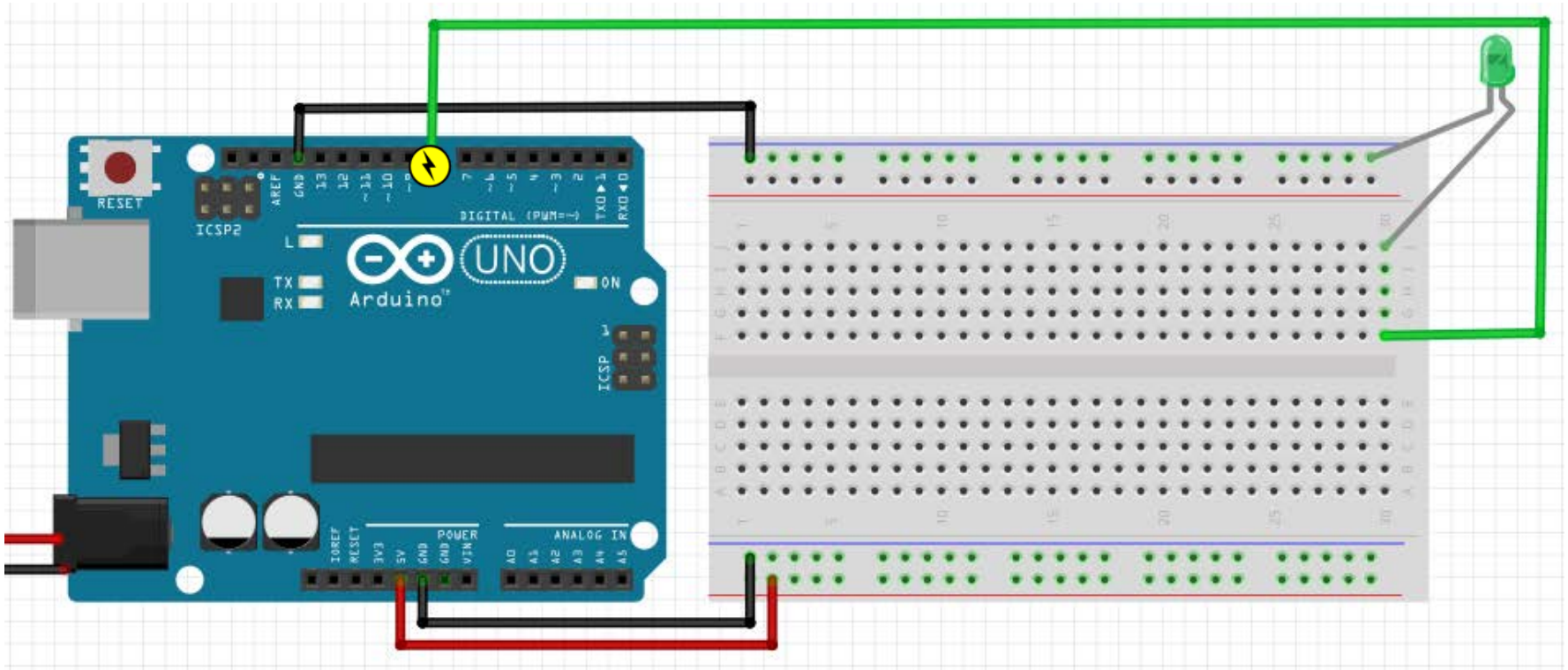
Step 3: Move the LED to Control the Light

Now we will move the light to the other side of the board. We will use pin #8 today, but any pin 2-13 could be used.



Step 3: Move the LED to Control the Light

Rather than be “always on”, this will allow us to control the light with a program we will install on the Arduino.

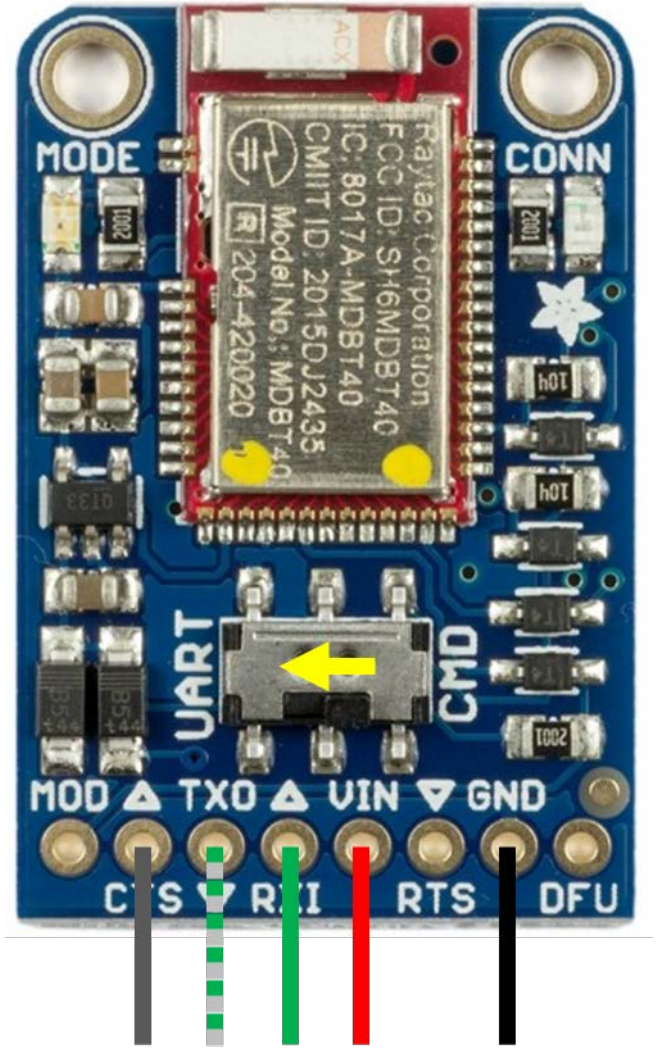


Step 4: Add Bluetooth Transmitter

- We chose this module because it has a ready-made app for both Android and iOS and doesn't require resistors with an Arduino.

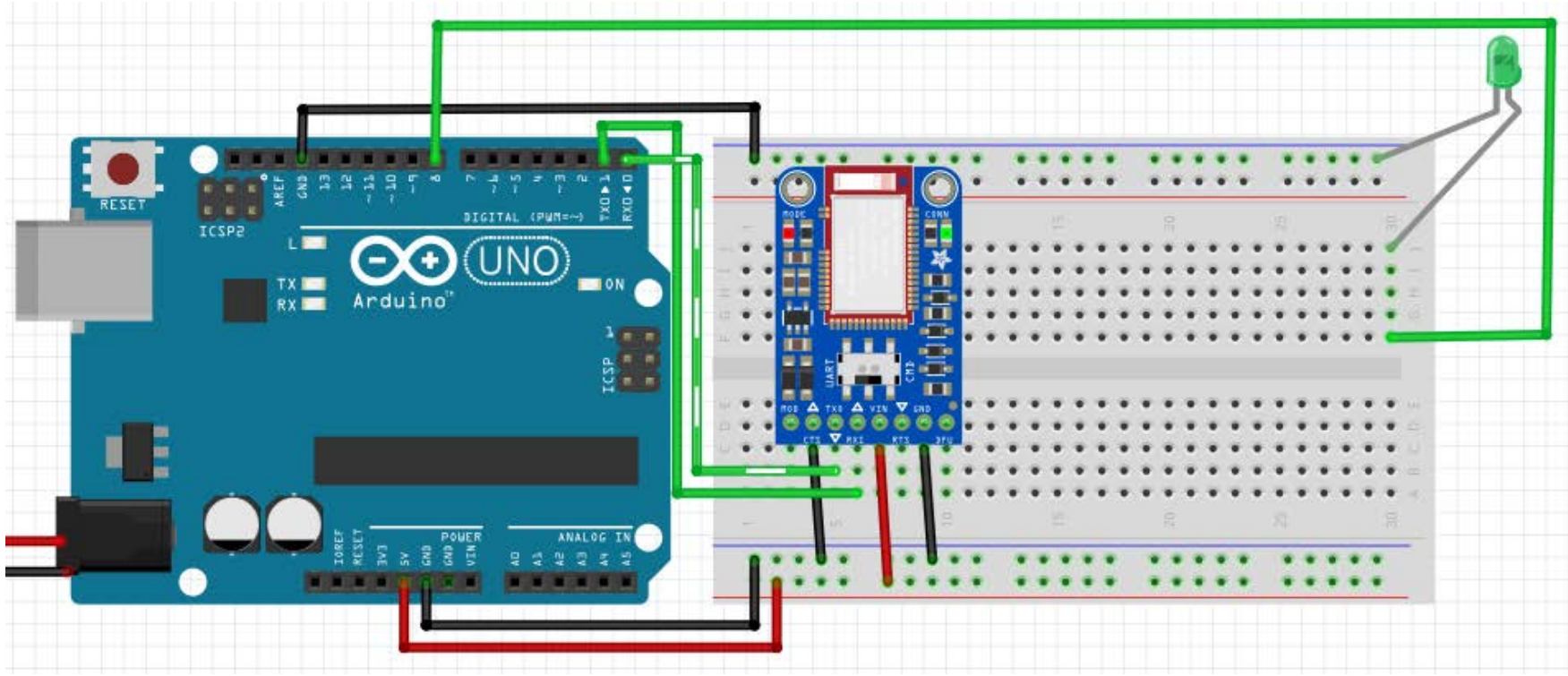
- There is a small physical switch on the module. Set it to Universal Asynchronous Receiver/Transmitter (UART).

- This module has several pins. The ones we will use are:
  - Voltage In (**VIN**) – Powers the radio
  - Ground (**GND**) – Powers the radio
  - Receive In (**RXI**) – Gets data from robot
  - Transmit Out (**TXO**) – Sends data to robot
  - Clear-to-Send (**CTS**) – Allows radio to control the robot



Step 4: Add Bluetooth Transmitter

The pins on the Bluetooth radio are designed to fit right into the breadboard. Connect the **power** and **signal** wires. Be sure to connect “Transmit” on the Arduino to “Receive” on the Bluetooth and vice versa.

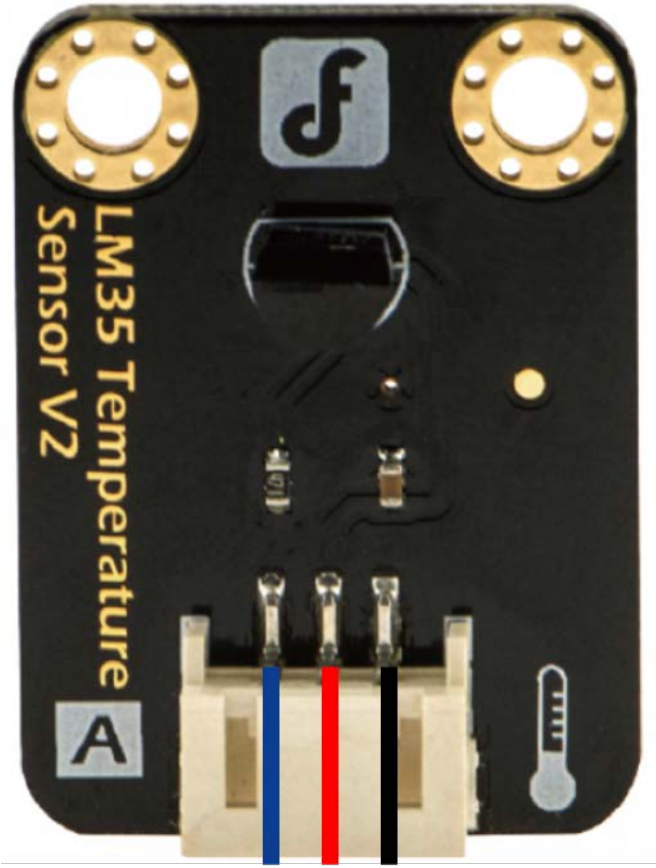


Step 5: Add the Temperature Sensor

- We chose this sample sensor because it is an easy to program and calibrate sensor.

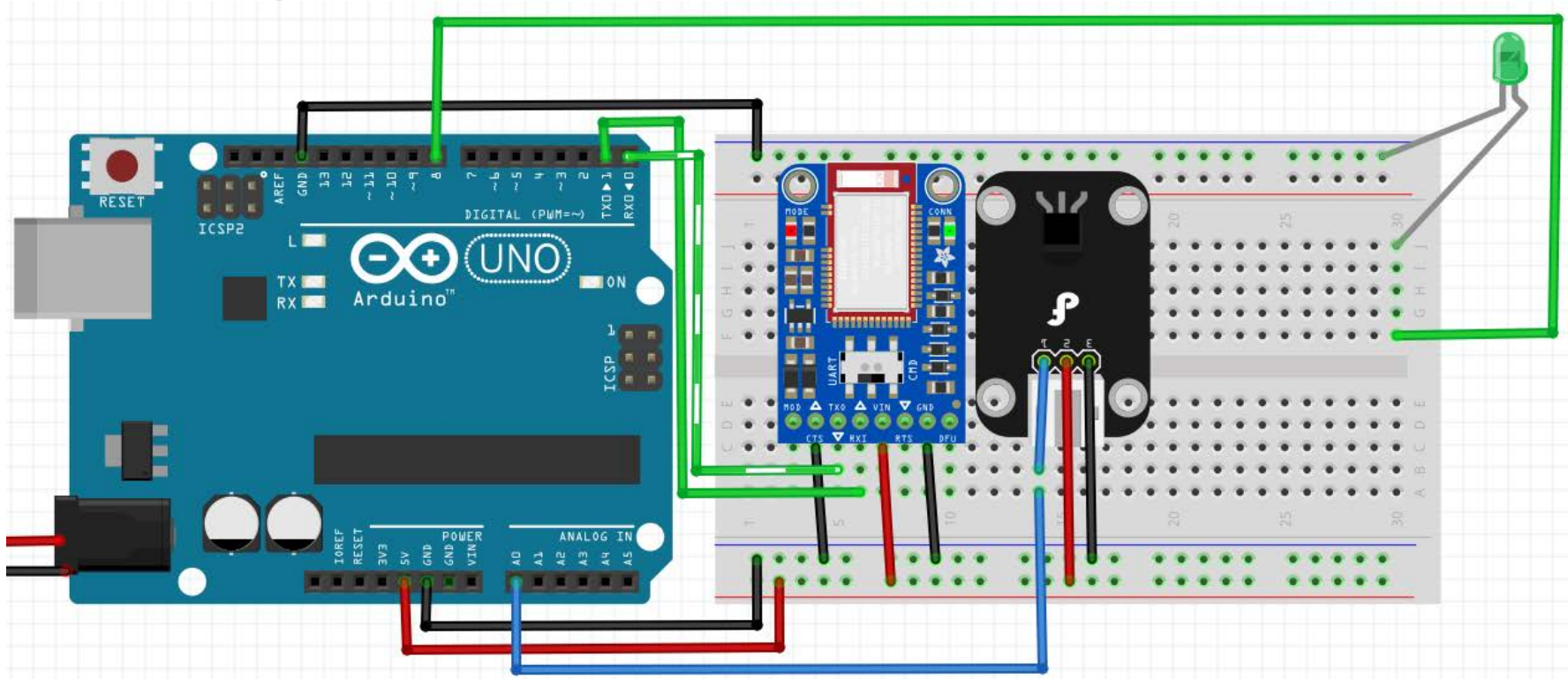
- It has only 3 pins:
  - **5V** – Power in
  - **GND** – Power out
  - **Output** – Analog signal out

- As the sensor changes temperature, it sends a different amount of voltage out of the **output** pin. We can convert this voltage to a Celsius temperature.



Step 5: Add the Temperature Sensor

The sensor has a ribbon that can be connected to the breadboard. Connect the **power** wires the same as before. Connect the **signal** wire to pin **A0**.

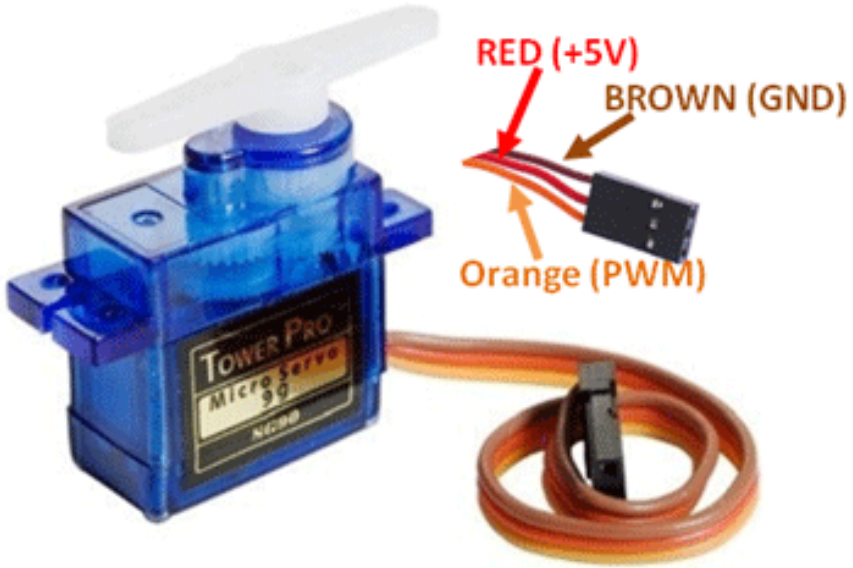


Step 6: Add a Motor to Deploy the Sensor

- Unless the data can be taken onboard the microcontroller, a motor will be needed to deploy the sensor to the surface.

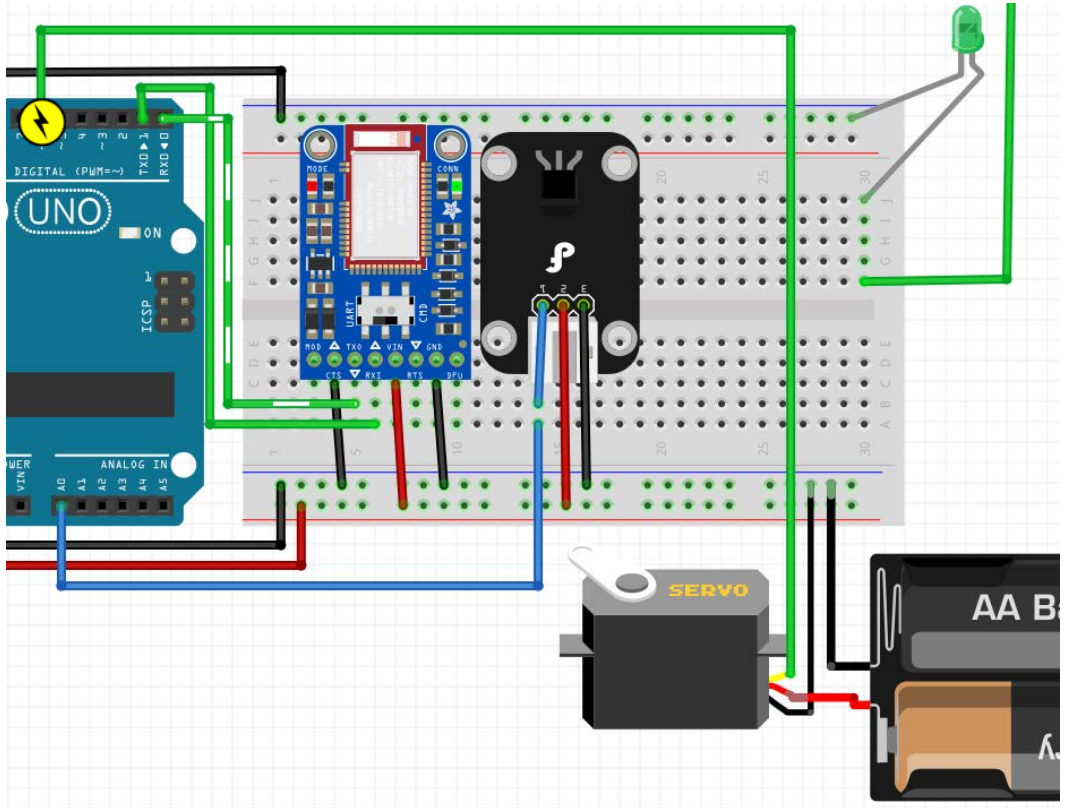
- It also has only 3 pins:
  - **5V** – Power in
  - **GND** – Power out
  - **Pulse Width Modulation (PWM)** – Controls the motor with a **digital** signal

- As the signal changes in duration, it controls the amount of rotation for the servo. We can use this movement to raise and lower an arm.

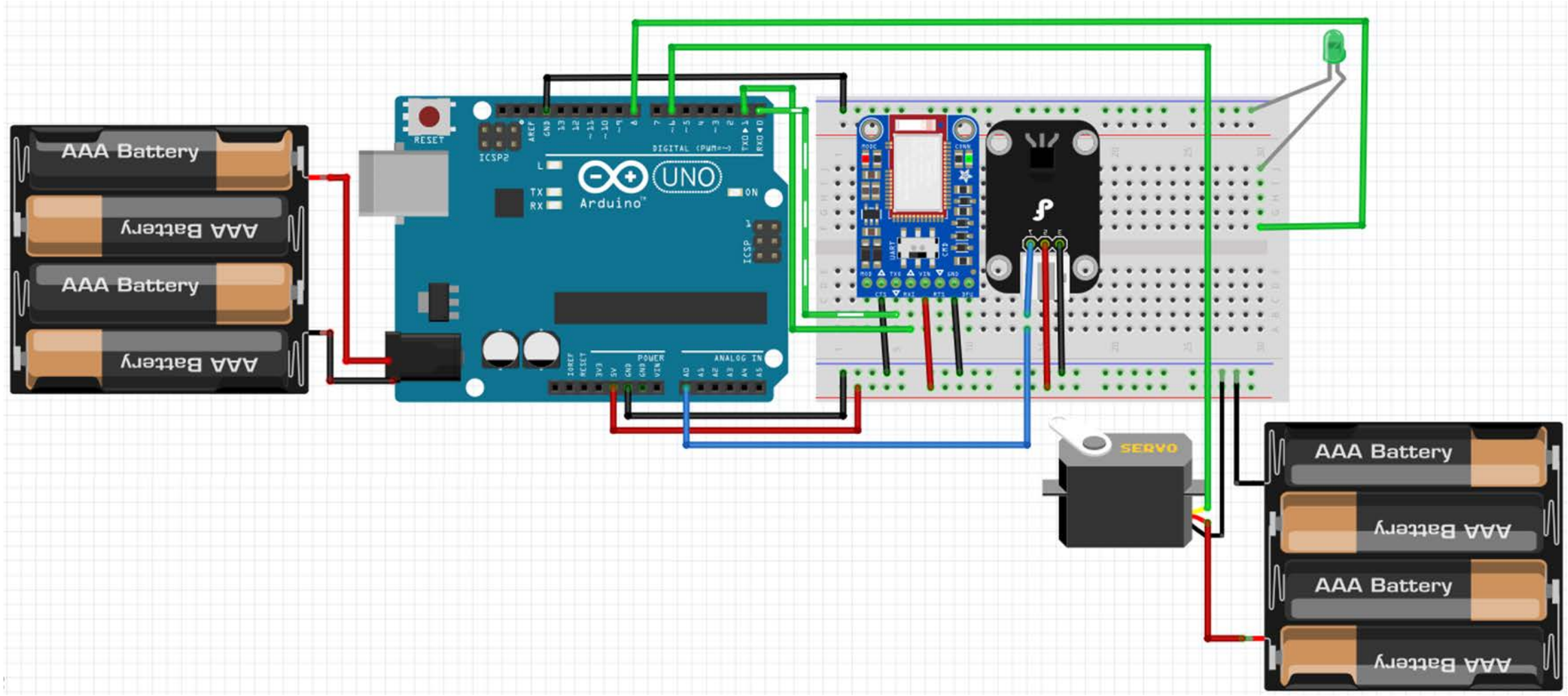


Step 6: Add a Motor to Deploy the Sensor

- The motor should be powered separately, so connect the **red** power to a battery pack. Connect the **PWM** wire to pin **D6** on the Arduino. The motor's and the battery's **GND** to the breadboard.



Complete Wiring Diagram



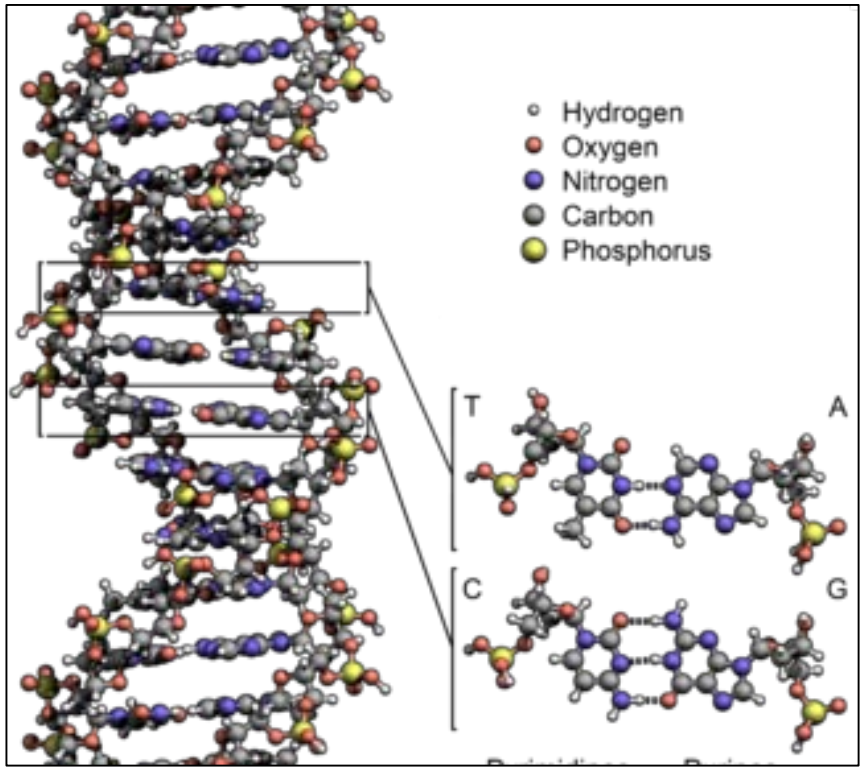
Programming the Sensor

- A program will follow the instructions it has been given.

- It does exactly what it has been instructed to do – no more, no less – even if those instructions are destructive to the system.

Programming the Sensor

Human



Computer

```
package de.wikibooks.javaee;

import javax.naming.InitialContext;
import javax.naming.NamingException;

public class StandaloneMain {

    /**
     * @param args
     */
    public static void main(String[] args) {
        StatelessSessionRemote myEJB = null;
        InitialContext ic;
        try
        {
            ic = new InitialContext();
            myEJB = (StatelessSessionRemote) ic.lookup("de.w

        } catch (NamingException e) {
            e.printStackTrace();
        }

        if (myEJB != null) {
            int result = myEJB.getAplusB(3, 4);
            System.out.println("Ergebnis: " + result);
```

Programming the Sensor: Blink Codes

It can be hard to tell what is going on inside the Arduino. Blinking an LED can be a simple way to know what is going on.

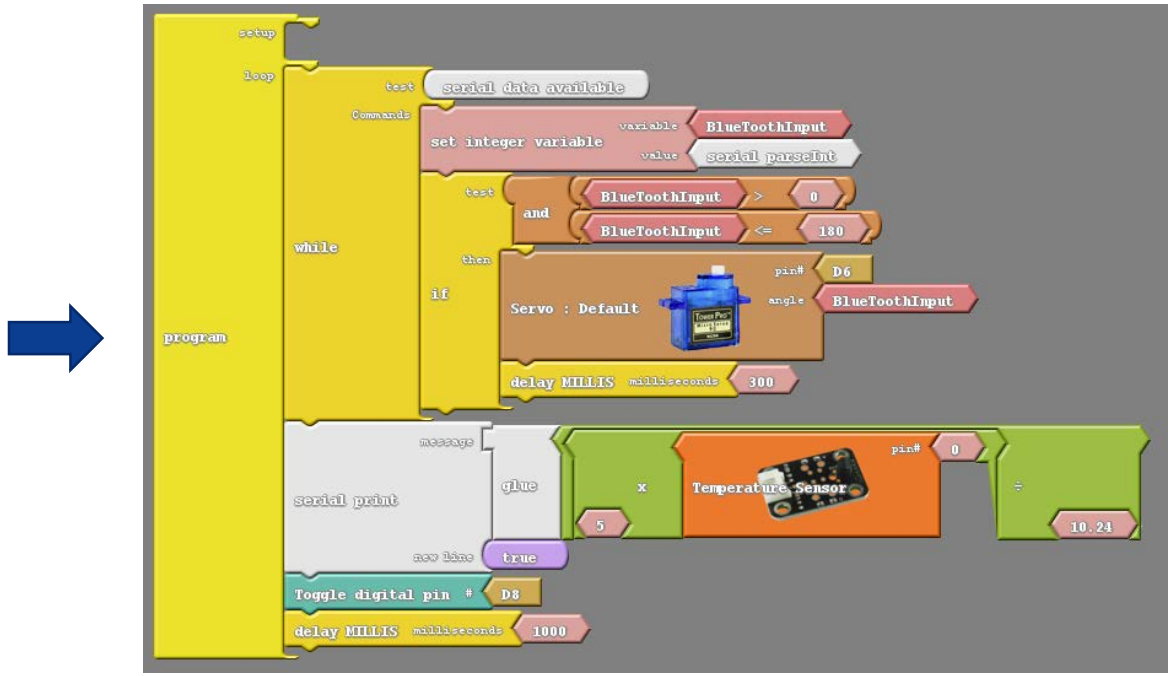
For example:

- On for a second, off for a second = “Everything is fine.”
- Two quick blinks = “Something important just happened.”
- Three long blinks = “Something went wrong.”

Programming the Sensor

Block code helps to visualize the sequence of code more easily than text.

```
#include <Servo.h>

int _ABVAR_1_BTInput = 0 ;
Servo servo_pin_8;
int __ardublockAnalogRead(int pinNumber)
{
  pinMode(pinNumber, INPUT);
  return analogRead(pinNumber);
}



void setup()
{
  Serial.begin(9600);
  servo_pin_8.attach(8);
}

void loop()
{
  while ( Serial.available() )
  {
    _ABVAR_1_BTInput = Serial.parseInt() ;
    if (( ( ( _ABVAR_1_BTInput ) >= ( 0 ) ) && ( ( _ABVAR_1_BT
    {
      servo_pin_8.write( _ABVAR_1_BTInput );
    }
  }

  Serial.print(map ( __ardublockAnalogRead(A0) , 0 , 1023 , 0
  //Serial.print(" ");
```



Block programming can’t do as much, but what it can do is easier!

Step 1: Install Coding Software

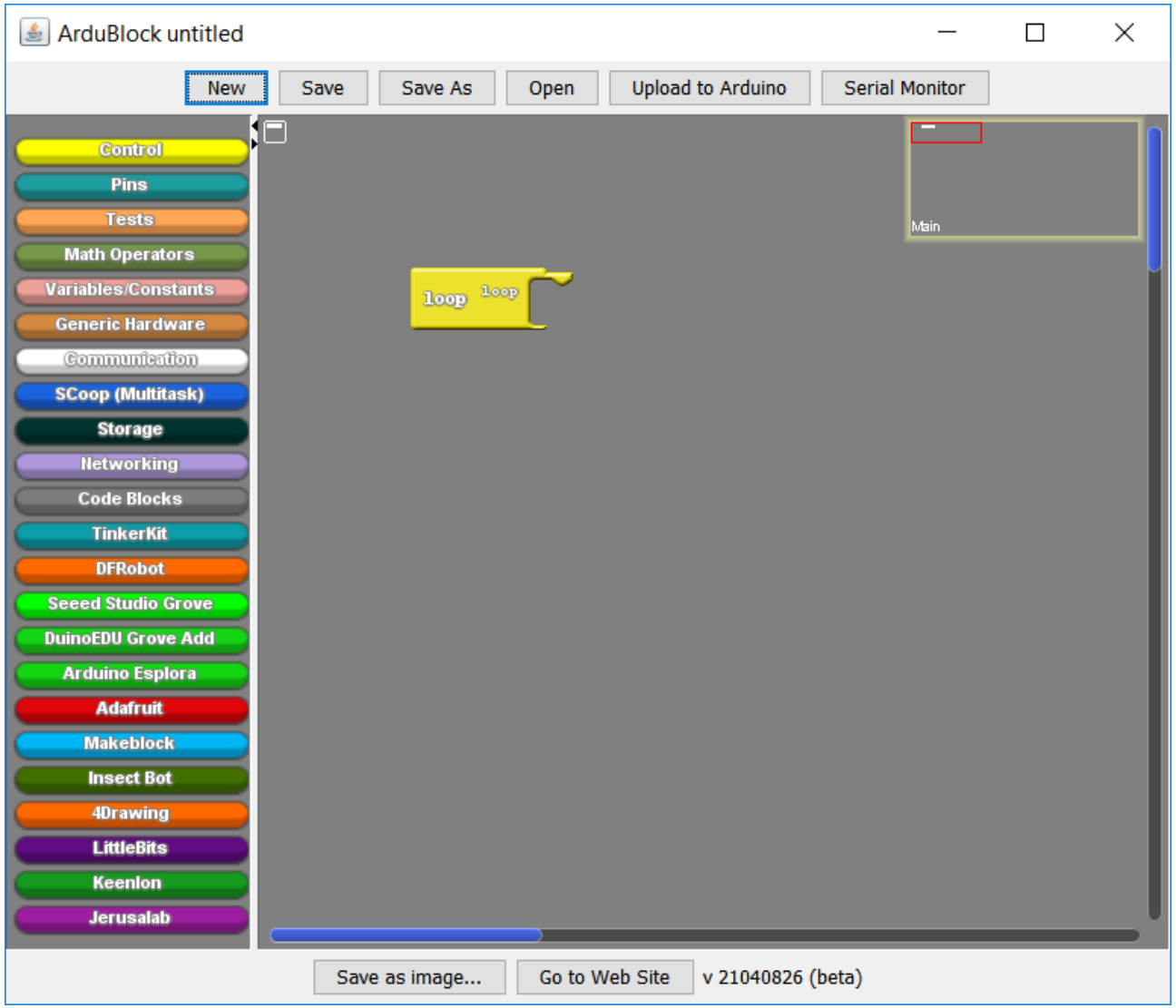
If you have not yet done so:

1. Download and install the Arduino software: https://www.arduino.cc/en/Main/Software

2. Download the latest version of Ardublock-all-master from here: https://github.com/taweili/ardublock/releases

3. Follow the Ardublock installation guide here: http://blog.ardublock.com/engetting-started-ardublockzhardublock/

The Ardublock Interface



The Ardublock Interface: Block Categories

Control - What to do, how often, and in what order

Pins – Working with digital output or analog inputs

Tests - Limits actions to certain conditions

Math Operators - Various arithmetic/ trigonometric calculations

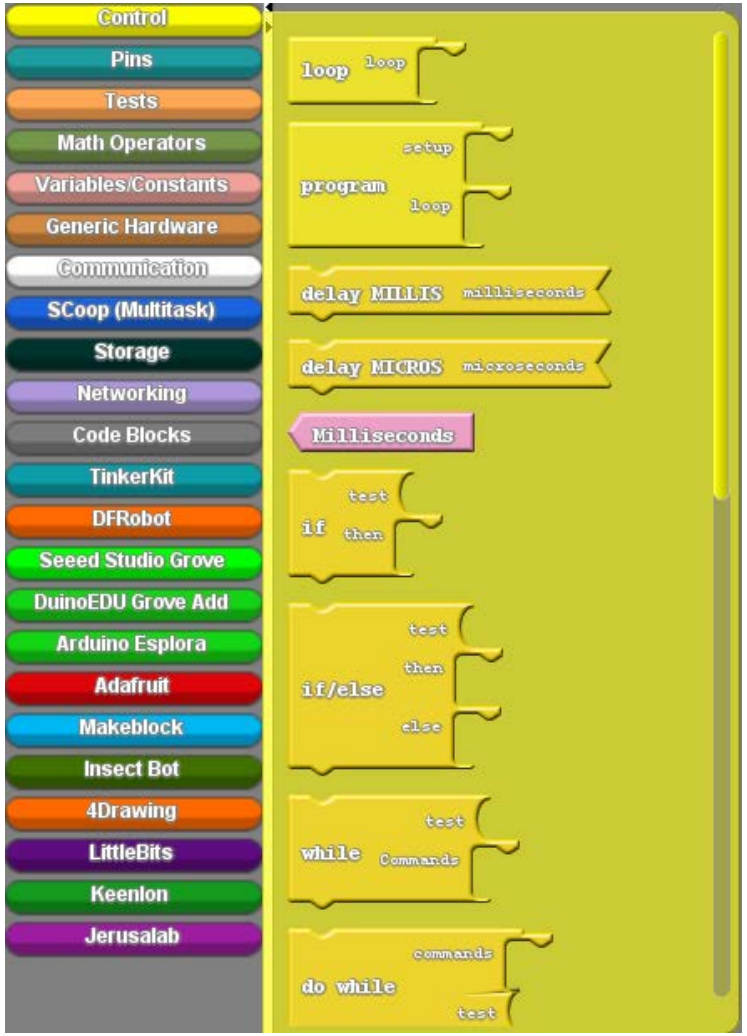
Variables/Constants - Storing, remembering, or changing values

Generic Hardware - Motors and other common hardware

Communication – Transmitting and receiving signals

Custom blocks to operate specific hardware from common suppliers.

The Ardublock Interface: Finding Blocks



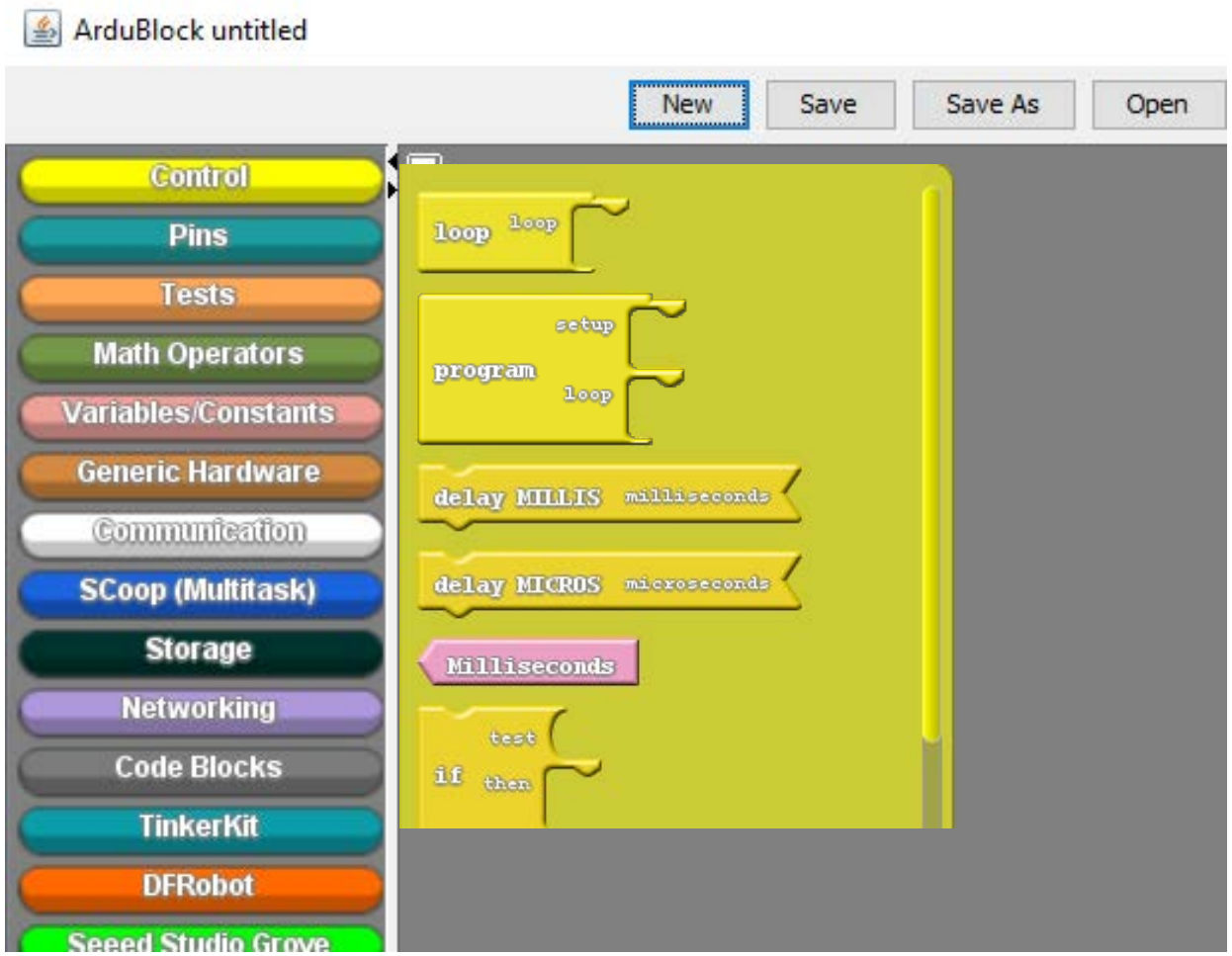
If you are looking for a yellow block, it is most likely in the yellow **Control** category. Likewise cyan blocks are most likely to be in the cyan **Pins** category

Note that not every block in the **Control** category is yellow.

Blocks with “sockets” on the right side will need another block that has a matching “plug” shape on the left side.

Some sockets can take more than one shape of plug. Experiment!

Step 2: Start Building the Program

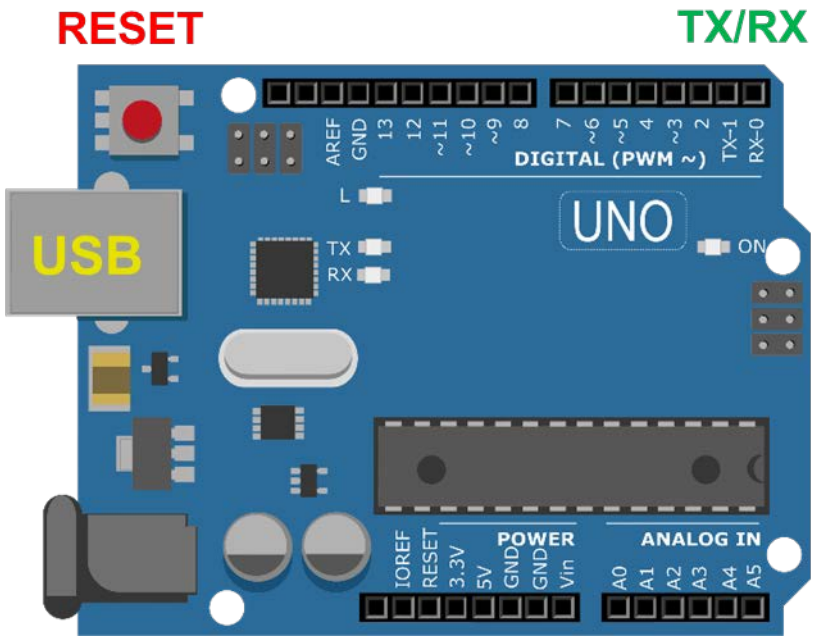


- By default the loop block is on every new Ardublock sketch.
- Drag off the “loop” block, then go to the **Control** menu, then drag the “program” block onto the sketch.
- The program block is the umbrella into which everything goes.
  - “Setup” happens once at the beginning
  - “Loop” happens over and over until the program is stopped.
- Blocks are processed top to bottom.

Step 3: The System Blinks Alive

- The “Toggle digital pin” block is found in the **Pins** category. It toggles a **Digital Output** from HIGH to LOW and vice-versa. Our LED was connected to pin **D8**.
- Normally, the loop runs as fast as the Arduino can manage. The “delay MILLIS” block is found in the **Control** category. It tells the program to wait for a set amount of time before doing anything else. Set it to **1000**.

Step 3: The System Blinks Alive

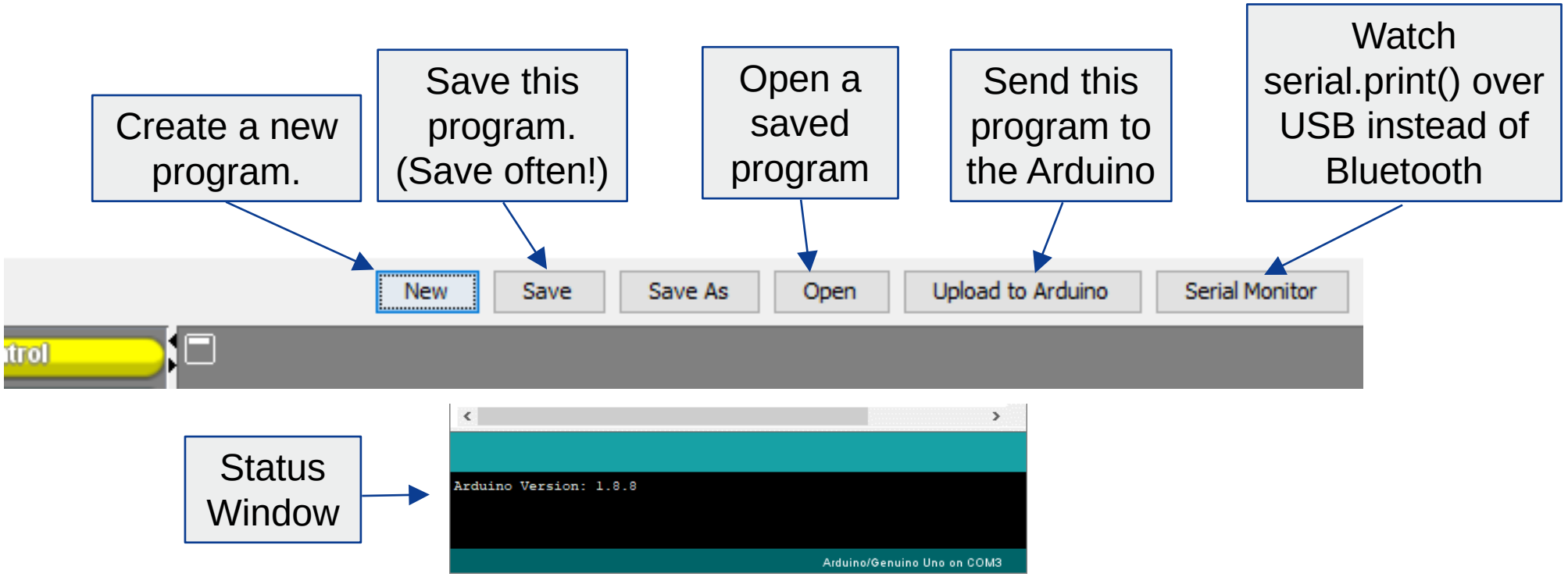


- Connect your Arduino to the computer via **USB** to upload your program. The computer can also power the Arduino through **USB** but not enough for motors or other higher-power hardware.
- The Arduino uses the same data connections to communicate with the computer via **USB** and anything connected via **TX/RX**. Temporarily disconnect the **TX/RX** wires when uploading code to the Arduino.
- Always reboot the Arduino with the **RESET** button after uploading a new program.

More info at: https://www.arduino.cc/en/Guide/ArduinoUno

Step 3: The System Blinks Alive

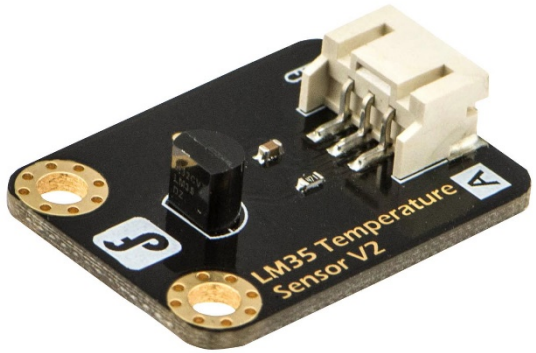
- When you “Upload to Arduino,” Ardublock will convert your block code to text code. Then, Arduino will attempt the upload.
- Status of the upload is listed at the bottom of the Arduino IDE window.
- Your LED should be slowly blinking now. If so, congratulations!



Step 4: Programming the Sensor

- Most sensors don’t actually use human measurements like °C

- The LM35 produces a number from 0 to 1023 based on the temperature and the maximum voltage available to it.

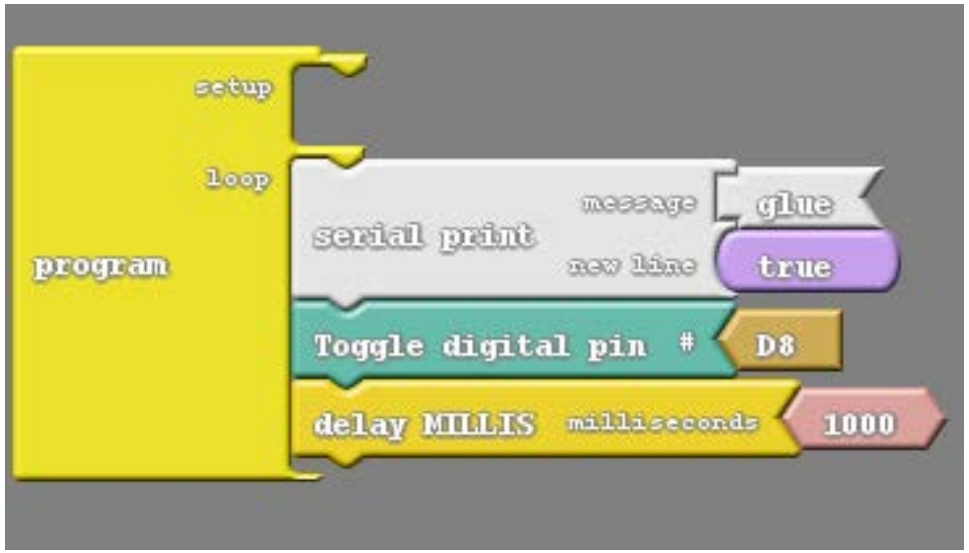


- In our case, 0°C reads 0 and 150°C reads 1023. We’ll need to code a conversion.

Step 4: Programming the Sensor

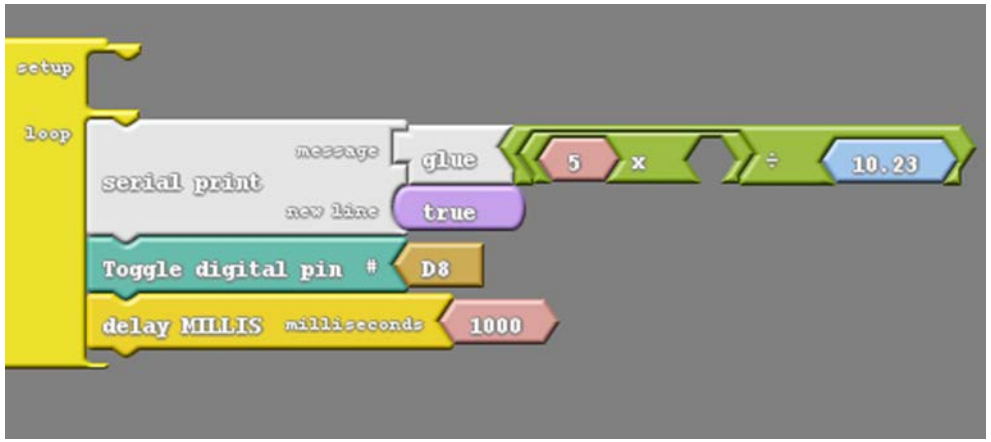
- Add a “serial print” from the Communication category.
- “Message” shows what string of characters are going to be sent (for now, through the **USB** cable).
- Pull off the extra “message” block.
- “New line” is asking if you want the message to end by moving to the next line.

Step 4: Programming the Sensor



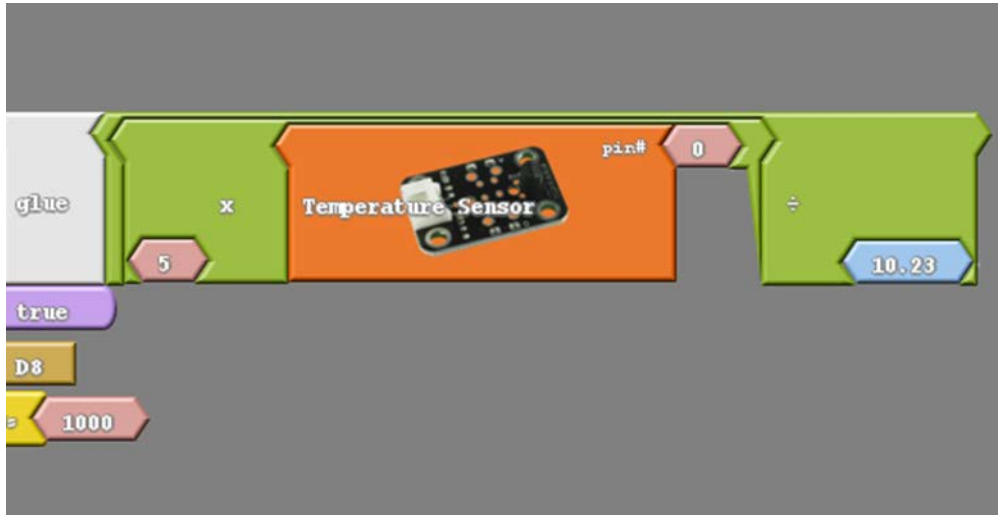
- Add a “glue” block from the Communication category.
- “Glue” allows you to send a number value as a series of characters in a message.

Step 4: Programming the Sensor



- Add two nested calculation blocks from the **Math Operators** category.
- Change their operations to “×” and “÷” respectively.
- Add two constant from the **Variables/Constants** category, one **integer** and one **decimal** constant.
- Change the values to 5 and 10.23 respectively.

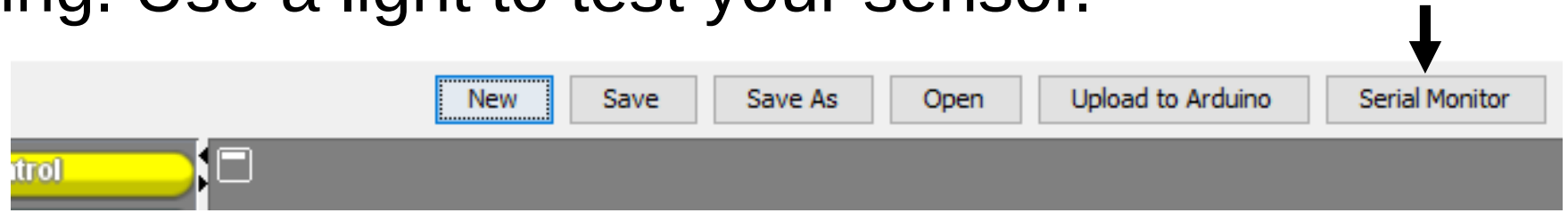
Step 4: Programming the Sensor

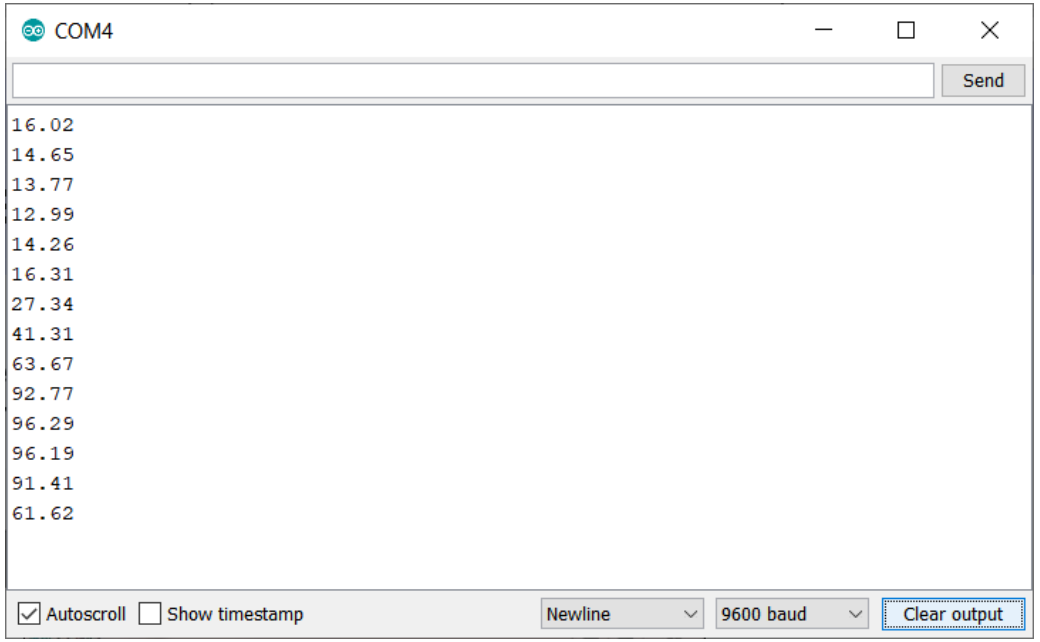


- Find the “Temperature Sensor” in the **DFRobot** category. Connect it to the × operation.
- Remember that we connected the sensor to analog pin# 0. Change the value if necessary.
- The program will now take the value from the sensor, multiply it by 5, then divide the result by 10.23 to get a temperature.

Step 4: Programming the Sensor
Real-time Output with Serial Monitor

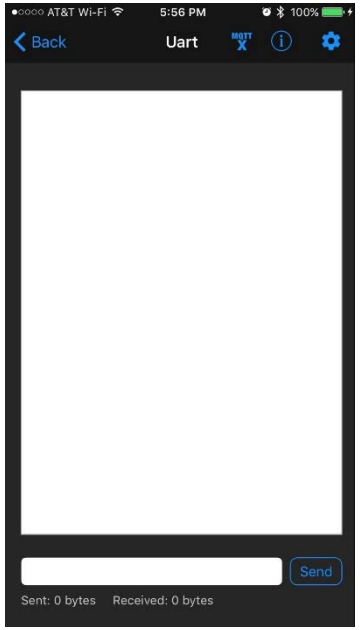
- Upload the latest program to your Arduino then click here to open the Serial Monitor. After a moment, numbers should start appearing. Use a light to test your sensor.

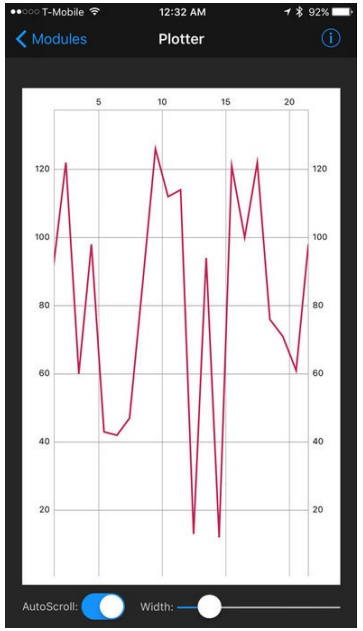




Step 5: Adding the Bluetooth Radio
Install the Bluefruit Connect App

The main features you'll want to explore are UART (which works like the Serial Monitor in the Arduino program) and the Serial Plotter (which graphs data received from the Arduino, if formatted properly)



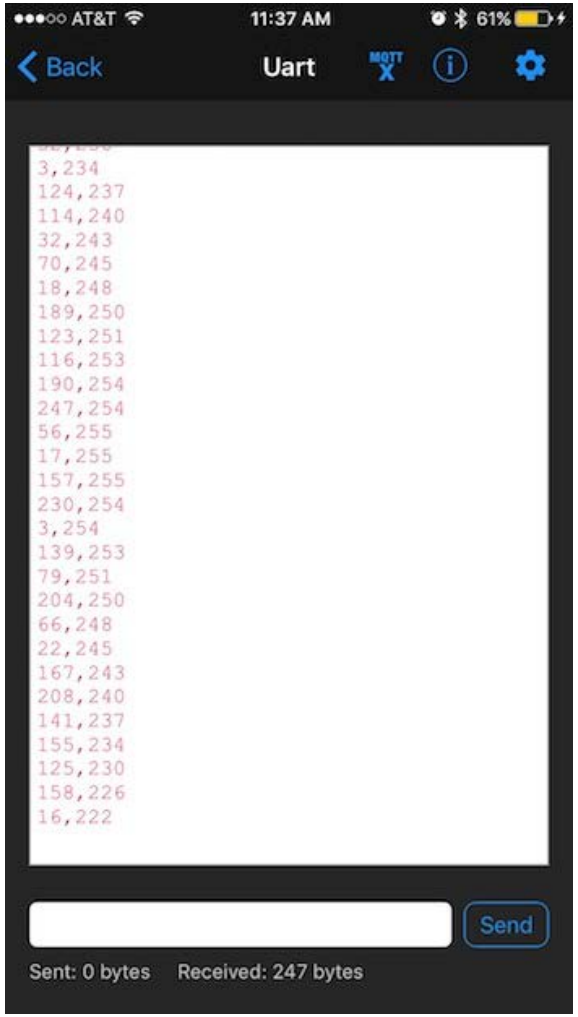


https://learn.adafruit.com/bluefruit-le-connect

Step 5: Adding the Bluetooth Radio

- Now you can receive the output from the temperature sensor straight on your phone!

- What if you wanted to control how far up or down the servo motor moved?

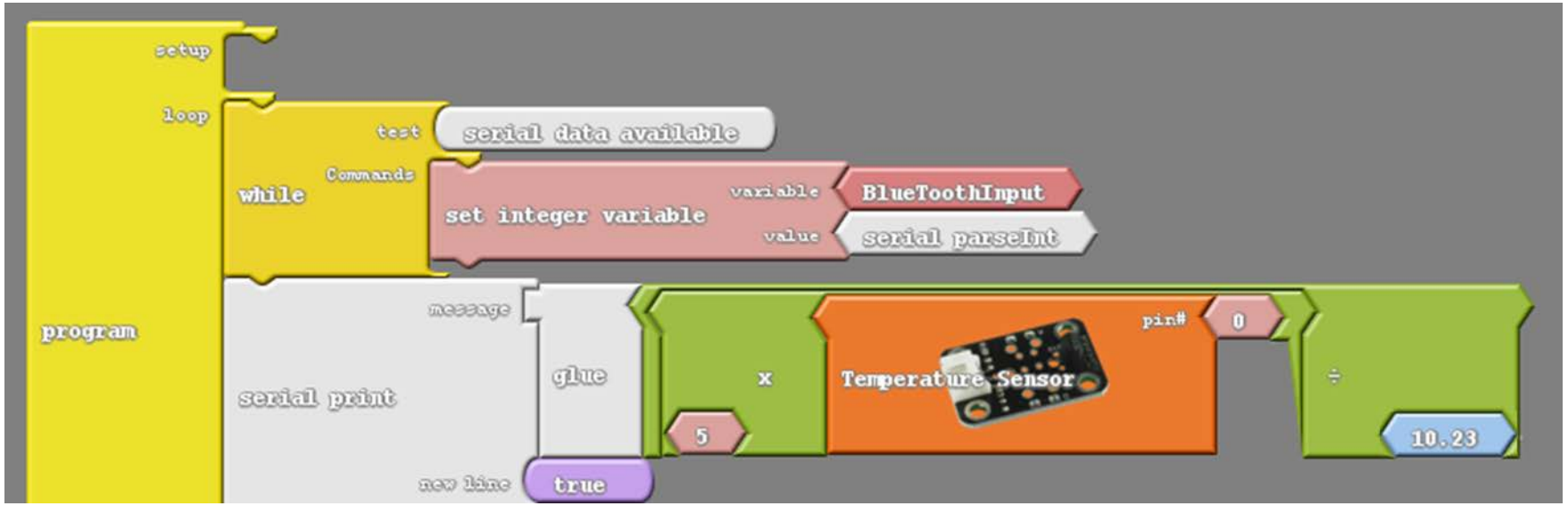


Step 5: Adding the Bluetooth Radio

Now we'll add the code to get input from the Bluetooth radio.

- Add a "while" block from the **Control** category. Use a "serial data available" from the Communication category. This will test if there is data coming from the Bluetooth radio.

- Add a "set integer variable" from the **Variables/Constants** category. Name the variable something helpful like "BlueToothInput" and the value must be set to the "serial parseInt" block.



Step 5: Adding the Bluetooth Radio

ASCII TABLE

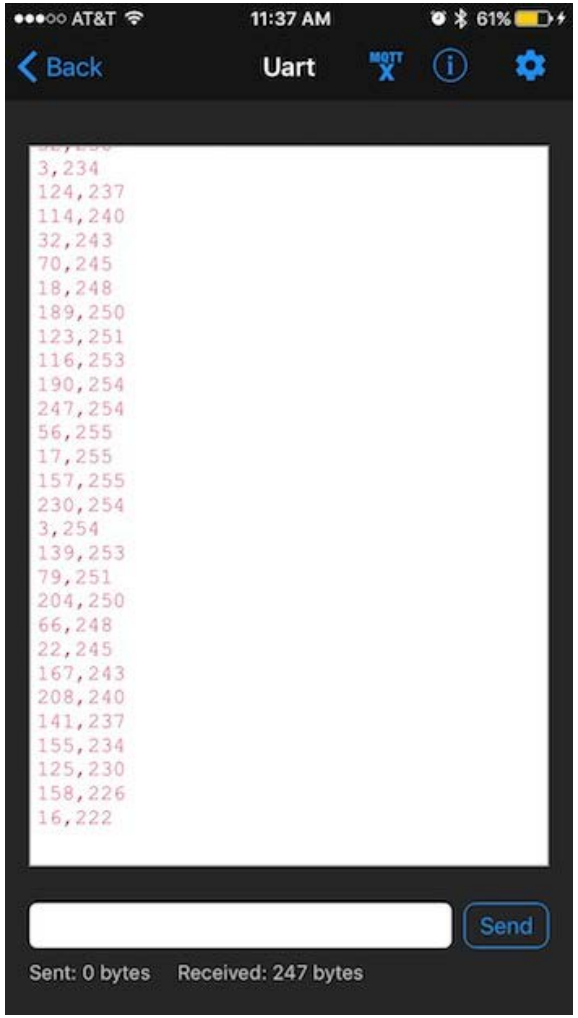
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

- Computers commonly use ASCII code to represent letters, numbers and symbols.
- The “serial read” block in Ardublock would see an input of “28” as “50” then “56” based on the decimal ASCII codes for 2 and 8, respectively.
- We need the “serial parseInt” block to read an input of “28” as an actual “28”.

Step 5: Adding the Bluetooth Radio

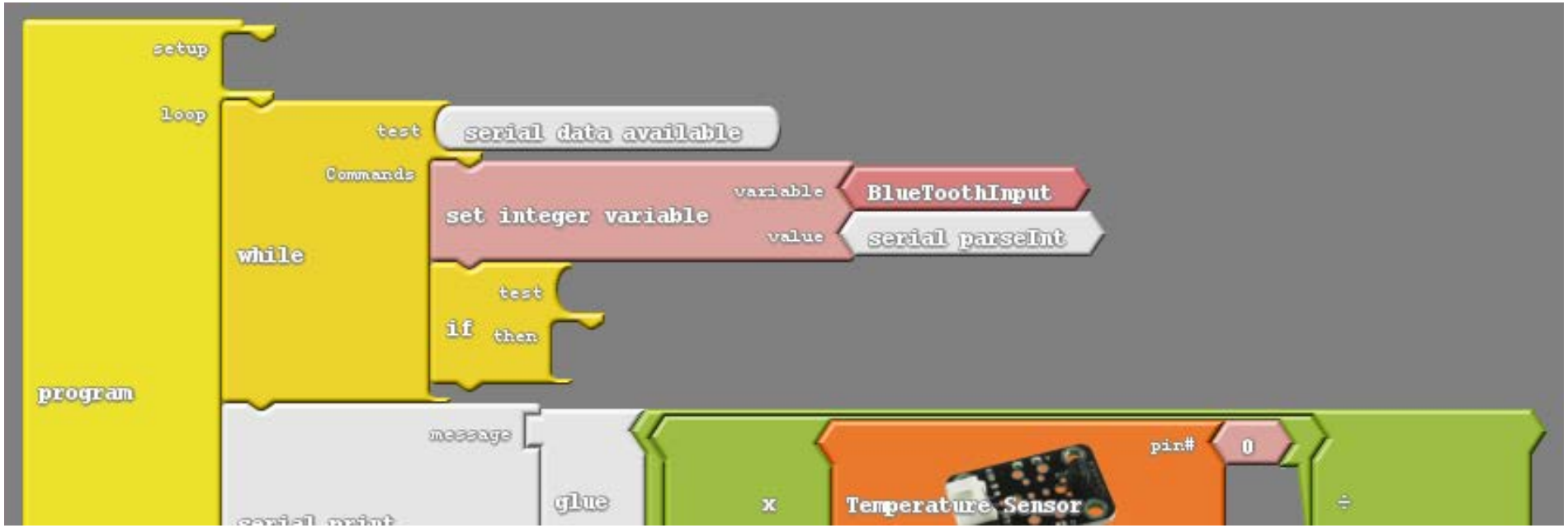
- By default, the serial input from your device will send a timeout “null” code after your last input.

- The computer will translate this to a “0”, so we need to account for that in the program.



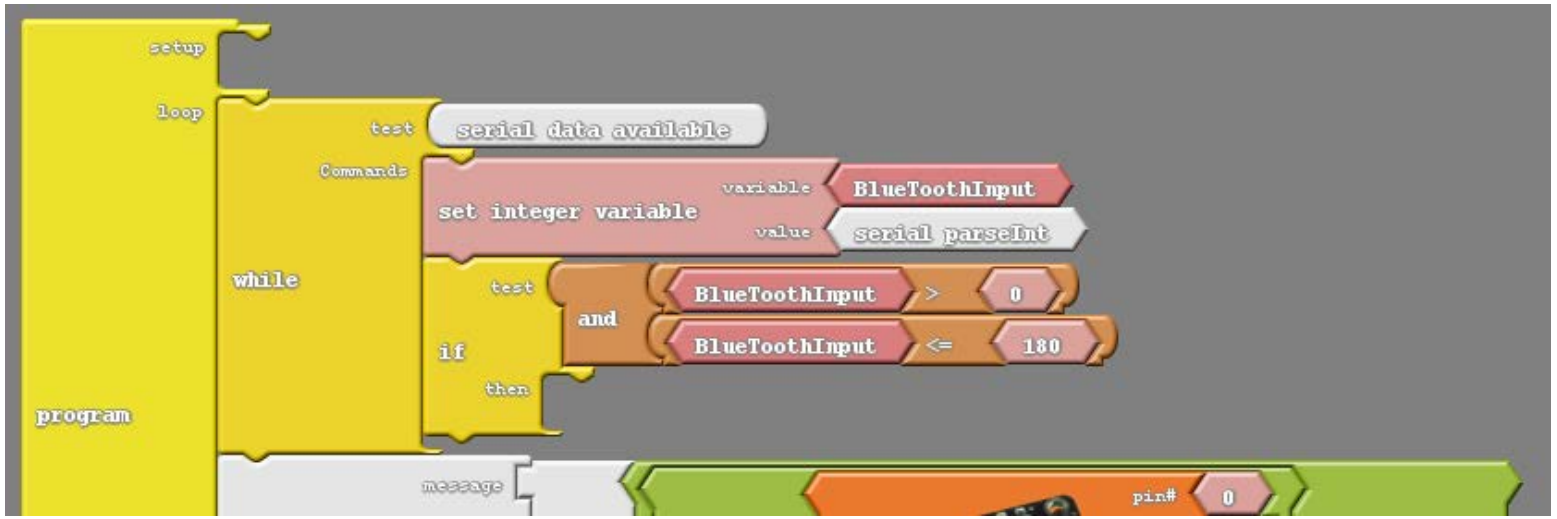
Step 6: Controlling the Motor

- Now we'll add commands to operate the motor that will control the sensor's arm.
- Start by adding an "if" block directly under the "set integer variable" we just added.



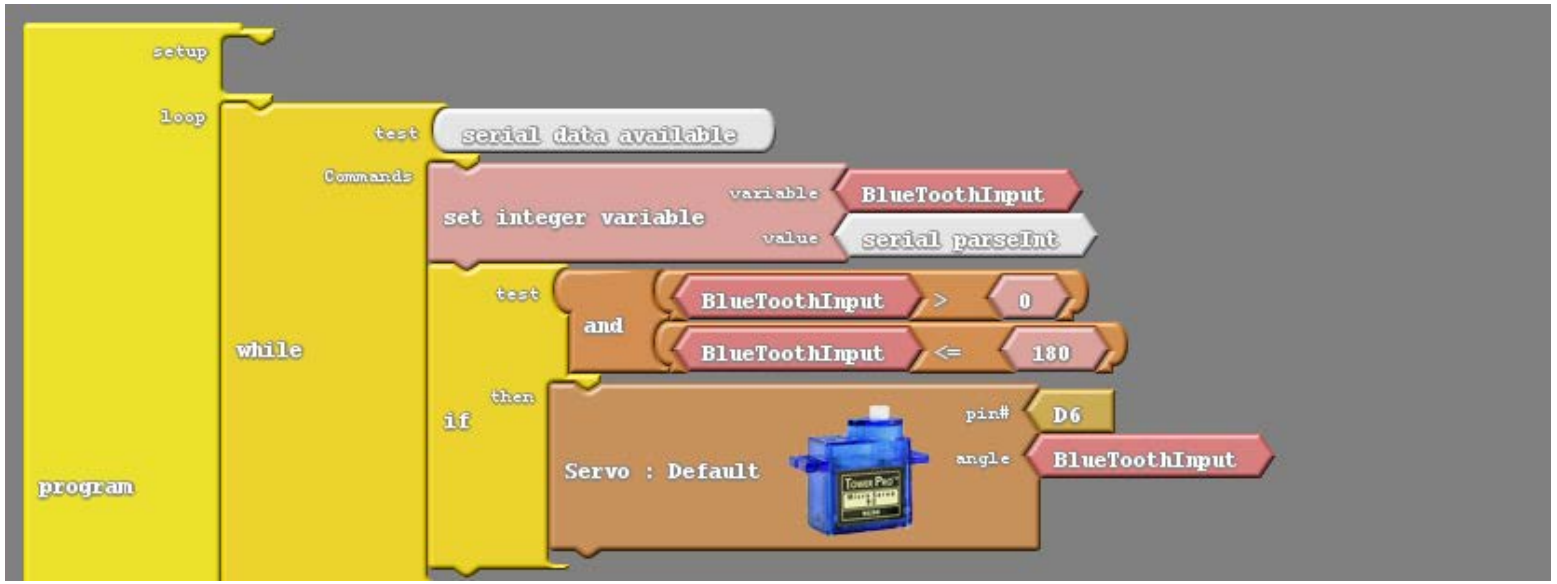
Step 6: Controlling the Motor

- Add an “and” block from the **Tests** category to the “if” block. This will allow us to test 2 criteria together.
- Add two comparison **tests**, “>” and “<=”
- Copy the Bluetooth **variable** you created before. Check if that **value** is >0 and <= 180.



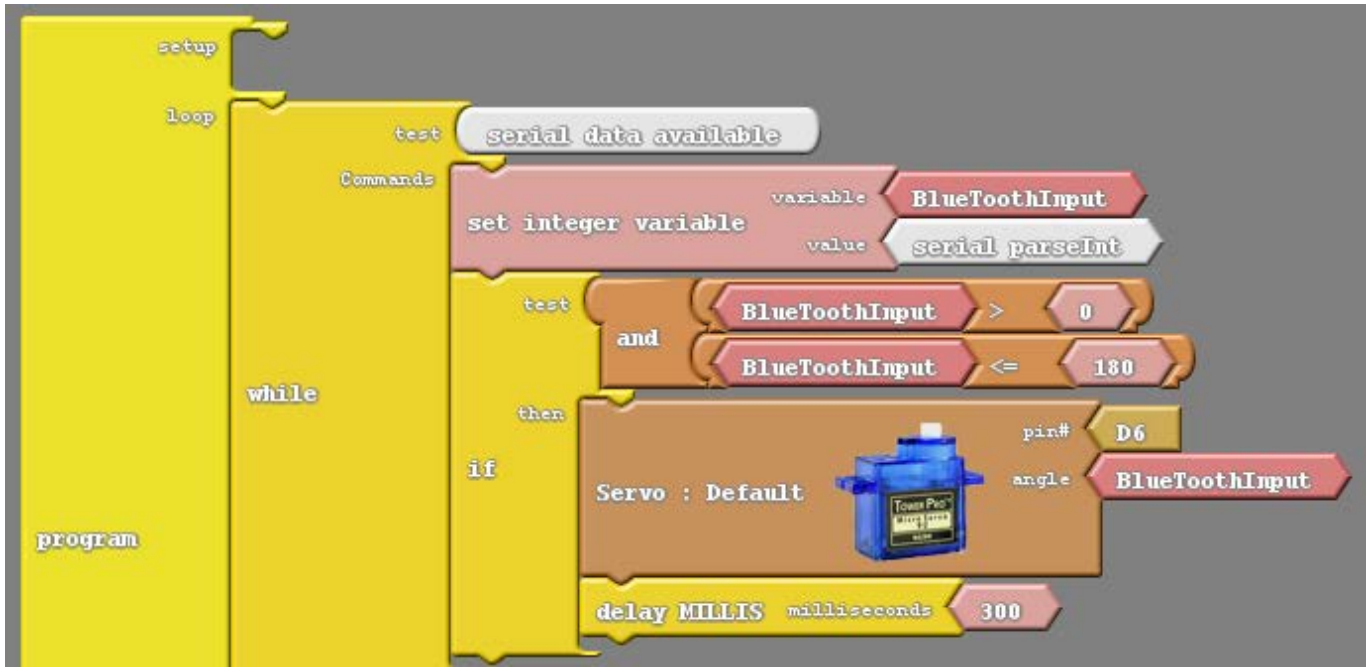
Step 6: Controlling the Motor

- Add a “Servo: Default” block from the **Generic Hardware** category to the “then” part of our “if” block.
- Remember we used digital pin# **D6** for our signal.
- Copy the Bluetooth **variable** one more time and use it as the angle of our arm (between 0° and 180°).

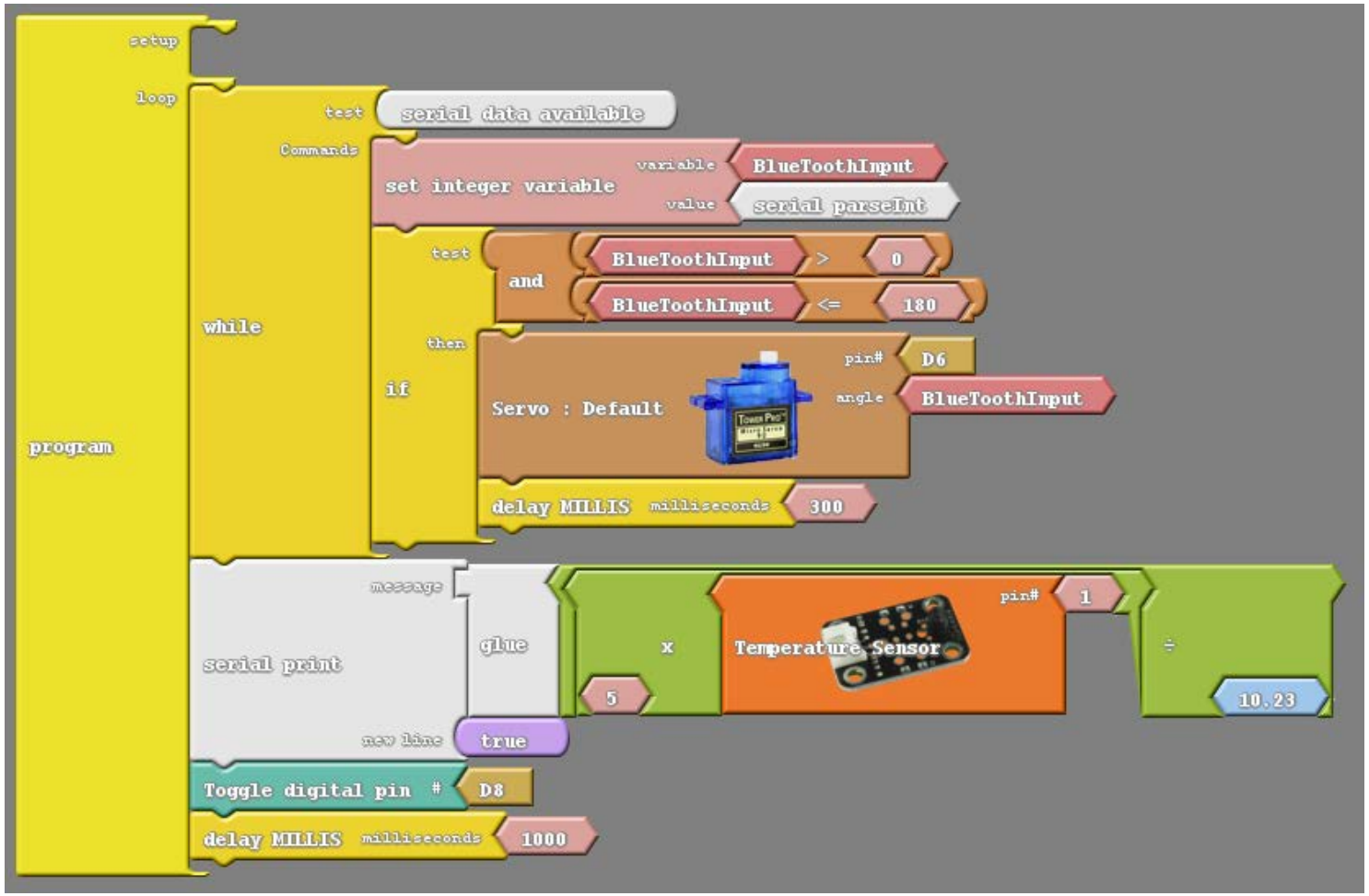


Step 6: Controlling the Motor

- Add a “delay MILLIS” block from the **Control** category after the motor command.
- Set the value to at least 100 milliseconds. This will give the motor time to move.



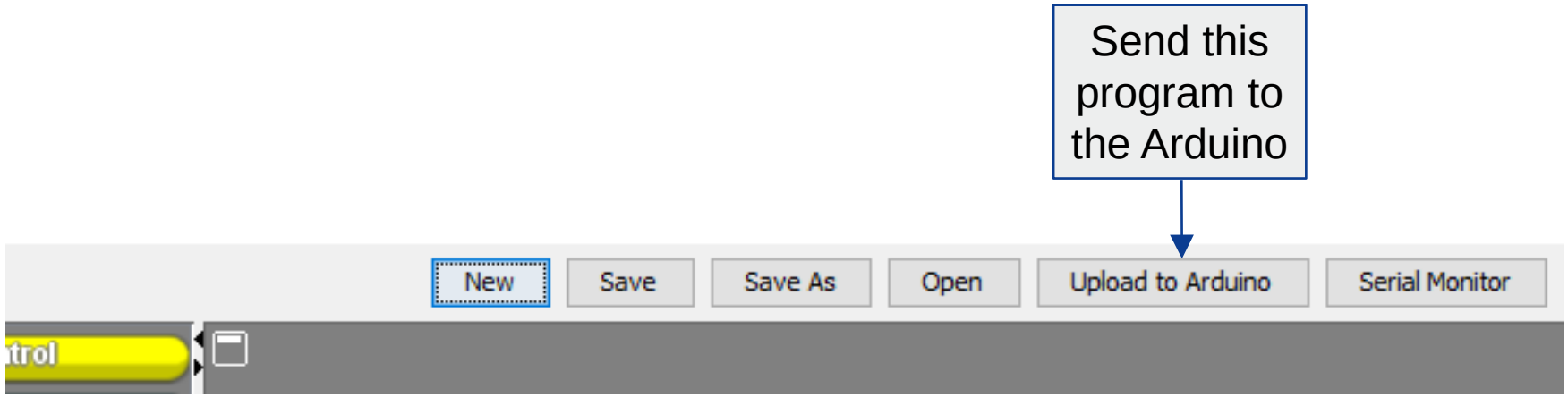
Step 6: Controlling the Motor



Congratulations! Your temperature sensor program is complete.

Step 6: Controlling the Motor
Testing Your Robot Components

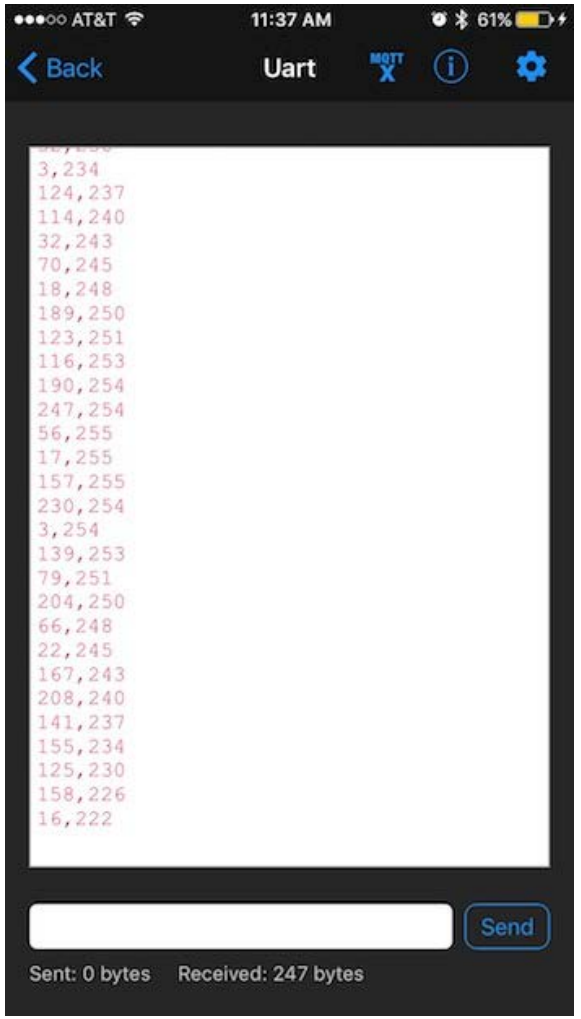
- Save your program and “Upload to Arduino”



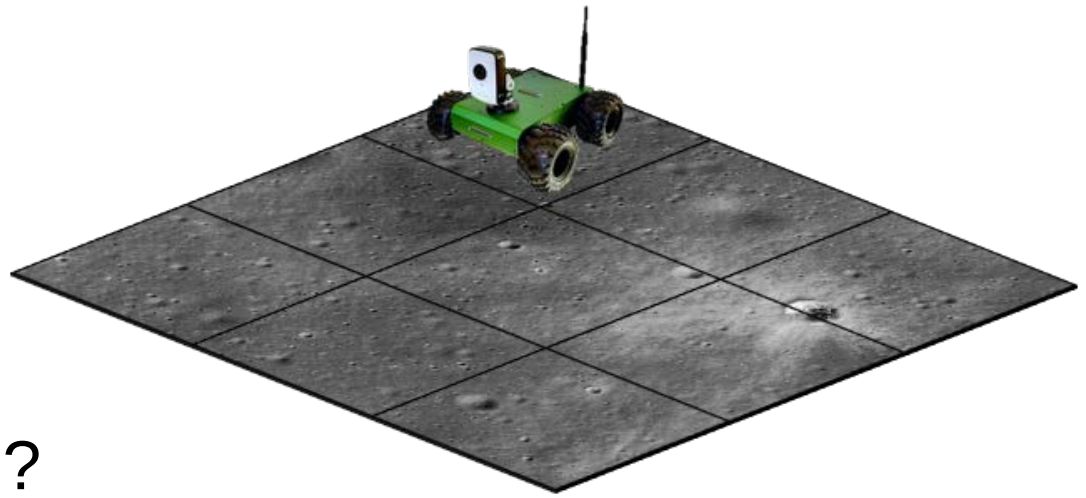
- You’ll also need to:
  - Disconnect the **USB**
  - Connect the external batteries and **TX/RX** wires.

Step 6: Controlling the Motor
Testing Your Robot Components

- Now you can send number values between 0 and 180 to control the motor, and you can receive the output from the temperature sensor on your phone!



- Now that you have all of the basic robot parts assembled and programmed. It will be up to the students to determine how to get them all on a single rover.
  - What type of rover will we use?
  - How will our payload fit on the rover?
  - How will we get data from various locations on the Moon's surface?
  - Do we need a payload deployment arm?
  - If so, how will it be designed?
  - What will our final data output look like?



Evolve

• **I tried something.** *How do I evolve it?*

- Look at your prototype, both the hardware and the code.
  - What parts you want to keep?
  - What parts can be improved?
  - Should anything be added?
  - Should anything be removed?

- Discuss your plan with other creators. What do they think could be improved?

Evolve

• **I tried something.** *How do I evolve it?*

- Make changes and take pictures of each finished model. These are called **iterations**.

- Your pictures should show how your model improved from the first to the last iteration. You will use these pictures on your presentation board

Share Your Story

- Create a presentation board to present your project.

- Be sure to include:
  - Your original problem
  - What you learned from research
  - Your brainstorm ideas
  - Your first sketch
  - Photos of your prototype
  - Notes from what you improved
  - Photos of all iterations
  - Your final solution model

- Practice presenting your project to others. Be ready to answer questions based on your experience.

Sample Making Project Rubric

	UNSATISFACTORY	COMPETENT	PROFICIENT	DISTINGUISHED
TECHNIQUE/ CONCEPTS	Work lacks understanding of concepts, materials and skills.	Work shows some understanding of concepts, materials and skills.	Work reflects understanding of concepts and materials, as well as use of skills discussed in class.	Work shows a mastery of skills and reflects a deep understanding of concepts and materials.
HABITS OF MIND	Student passively attempts to fulfill assignment without much thought or exploration of possibilities. Student refuses to explore more than one idea.	Developing exploration of possible solutions and innovative thinking. Student has more than one idea but does not pursue.	Student explores multiple solutions and innovative thinking develops and expands during project.	Consistently displays willingness to try multiple solutions and ask thought provoking questions, leading to deeper, more distinctive results. Student fully explores multiple ideas and iterations.
REFLECTION & UNDERSTANDING	Student shows little awareness of their process. The work does not demonstrate understanding of content.	Student demonstrates some self-awareness. Work shows some understanding of content, but student cannot justify all of their decisions.	Student shows self-awareness. Work demonstrates understanding of content and most decisions are conscious and justified.	Work reflects a deep understanding of the complexities of the content. Every decision is purposeful and thoughtful.
CRAFTSMANSHIP	Work is messy and craftsmanship detracts from overall presentation.	Work is somewhat messy and craftsmanship detracts somewhat from overall presentation.	Work is neat and craftsmanship is solid.	Work is impeccable and shows extreme care and thoughtfulness in its craftsmanship.
RESPONSIBILITY	Frequent illegal absences, tardiness, disrespect for classmates and teacher. Disregard for materials and work such as refusal to clean up or throwing out work.	Student is sometimes illegally absent, tardy, or disrespectful. Must be persuaded to assist in clean up and to take work home.	Student is most often present, on time, and respectful. Usually participates willingly in clean up and takes pride in work.	Student is consistently present, punctual, and respectful of classmates and teacher. Self-directed clean up and ownership of work.
EFFORT	Work is not completed in a satisfactory manner. Student shows minimal effort. Student does not use class time effectively.	Work complete but it lacks finishing touches or can be improved with a little effort. Student does just enough to meet requirements.	Completed work in an above average manner, yet more could have been done. Student needs to go one step further to achieve excellence.	Completed work with excellence and exceeded teacher expectations. Student exhibited exemplary commitment to the project.

Created by Lisa Yokana, https://www.edutopia.org/blog/creating-authentic-maker-education-rubric-lisa-yokana

Thank You for Participating!