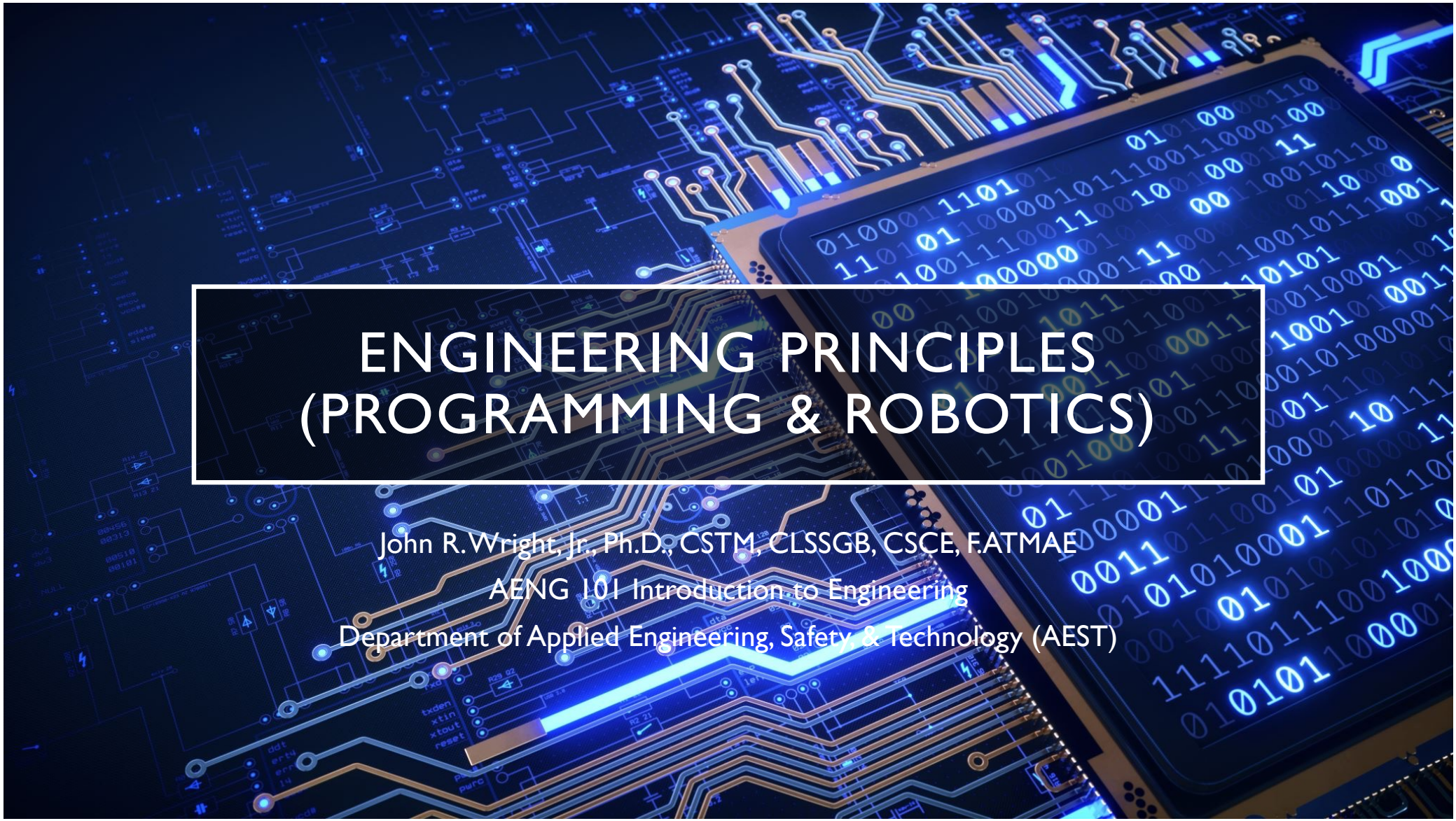


ENGINEERING PRINCIPLES (PROGRAMMING & ROBOTICS)

John R. Wright, Jr., Ph.D., CSTM, CLSSGB, CSCE, F.ATMAE
AENG 101 Introduction to Engineering
Department of Applied Engineering, Safety, & Technology (AEST)

OUTLINE

What is a Microcontroller?

- History of the microcontroller
- What is an IDE?

High-level languages

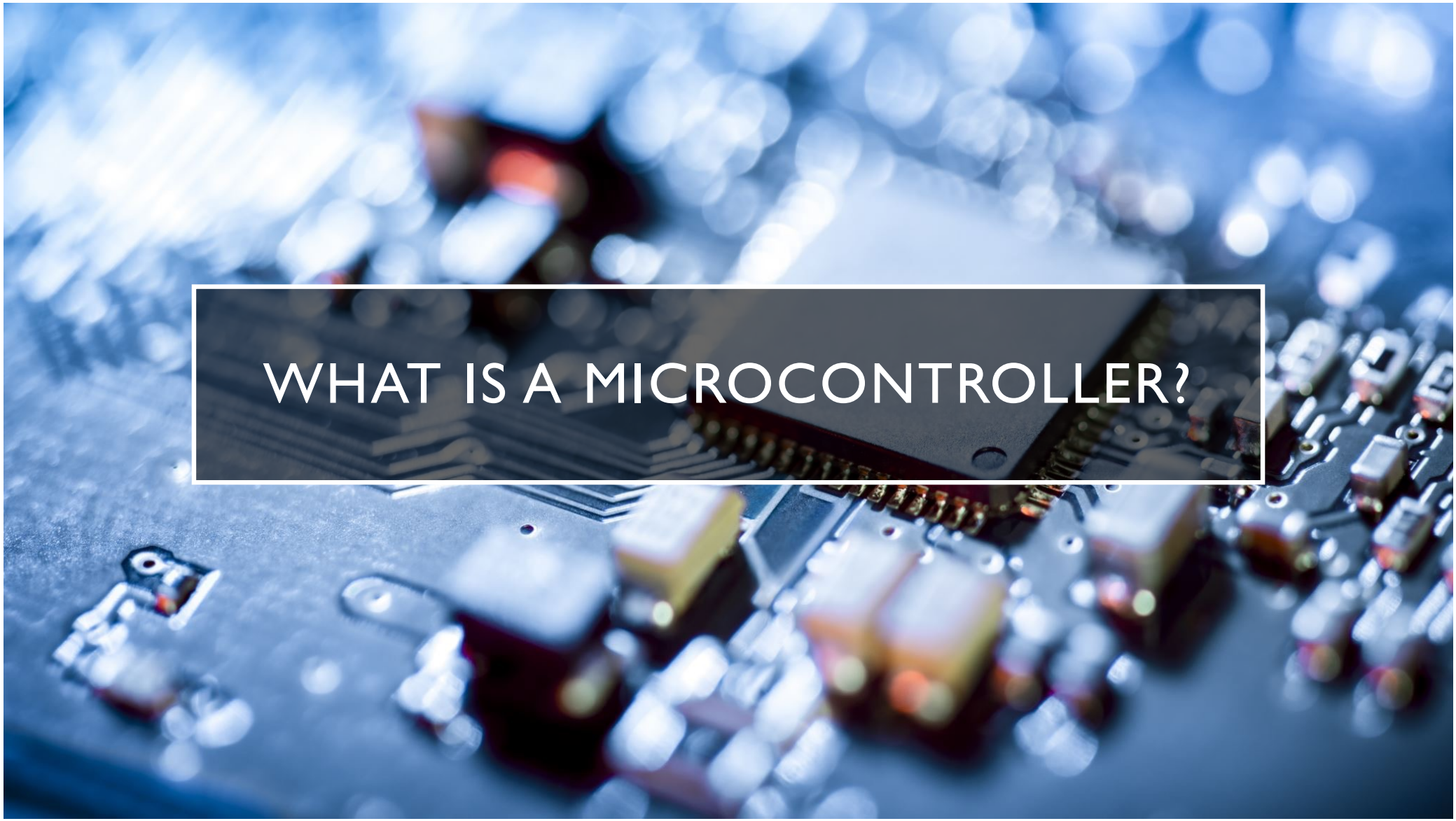
Code Fundamentals (C++)

- Variables
- Sensors / inputs
- Functions
- Libraries
- Servo motor control / outputs
- If Statements

Algorithms, Pseudocode, & Code

Flowcharts

Real-time I/O


WHAT IS A MICROCONTROLLER?

BRIEF HISTORY OF THE MICROCONTROLLER

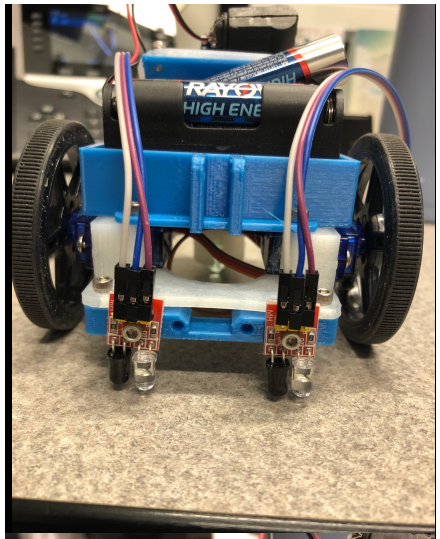
- In 1971, the first microcontroller was invented by two engineers at Texas Instruments, according to the Smithsonian Institution.
  - Gary Boone and Michael Cochran created the TMS 1000, which was a 4-bit microcontroller with built-in ROM and RAM.
  - The same year that the microprocessor was invented at Intel
- The microcontroller was used internally at TI in its calculator products from 1972 until 1974, and was refined over the years.
- In 1974, TI offered the TMS 1000 for sale to the electronics industry. The TMS 1000 was available in various configurations of RAM and ROM sizes.

http://www.ehow.com/info_10018768_history-microcontroller.html

BRIEF HISTORY OF THE MICROCONTROLLER

- During the 1990s, microcontrollers with electrically erasable and programmable ROM (EEPROM) memories, such as flash memory, became available.
  - These microcontrollers could be programmed, erased and reprogrammed using only electrical signals.
  - Prior to the electrically reprogrammable devices, microcontrollers often required specialized programming and erasing hardware, which required that the device be removed from its circuit, slowing software development and making the effort more expensive.
  - With this limitation removed, microcontrollers were able to be programmed and reprogrammed while in a circuit so devices with microcontrollers could be upgraded with new software without having to be returned to the manufacturer. Many current microcontrollers, such as those available from Microchip and Atmel, incorporate flash memory technology.

http://www.ehow.com/info_10018768_history-microcontroller.html



Arduino Nano V3.0 Microcontroller Board

ATmega328 CH340G

Digital I/O Pins: 14 (of which 6 provide PWM output)

Operating Voltage: 5V

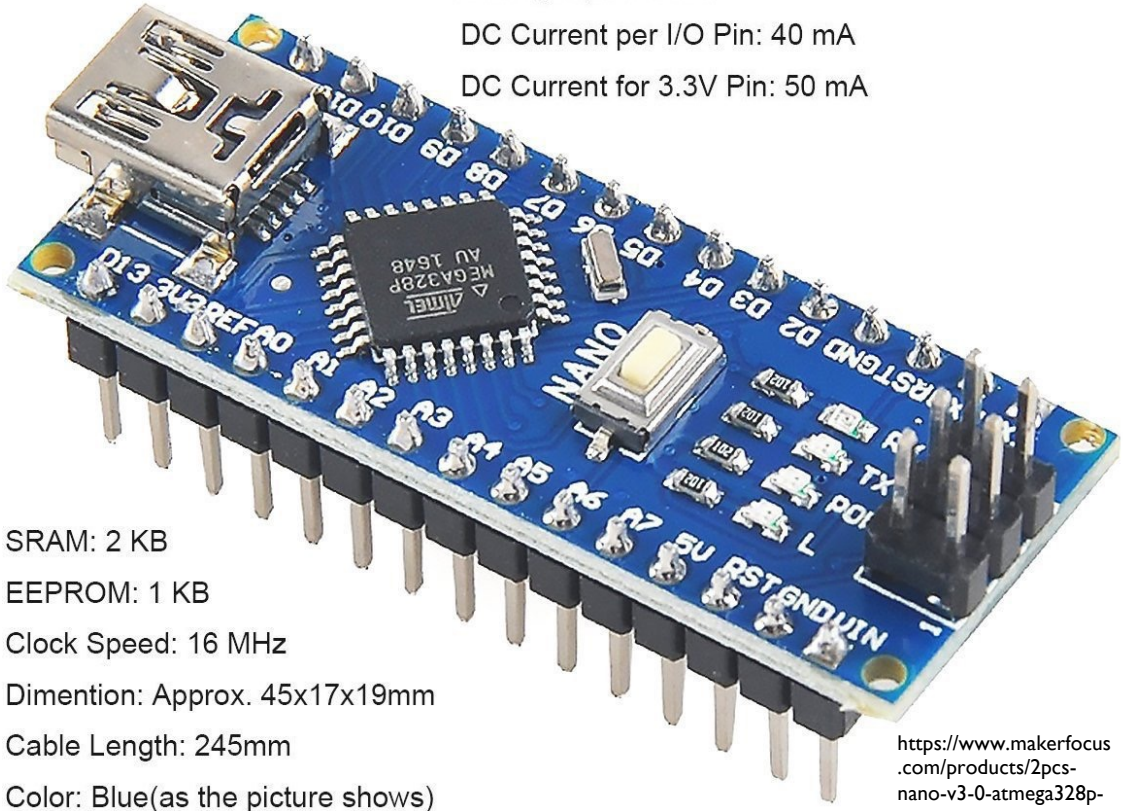
Input Voltage: (recommended) 7V-12V

Input Voltage: (limits) 6V-20V

Analog Input Pins: 8

DC Current per I/O Pin: 40 mA

DC Current for 3.3V Pin: 50 mA



SRAM: 2 KB

EEPROM: 1 KB

Clock Speed: 16 MHz

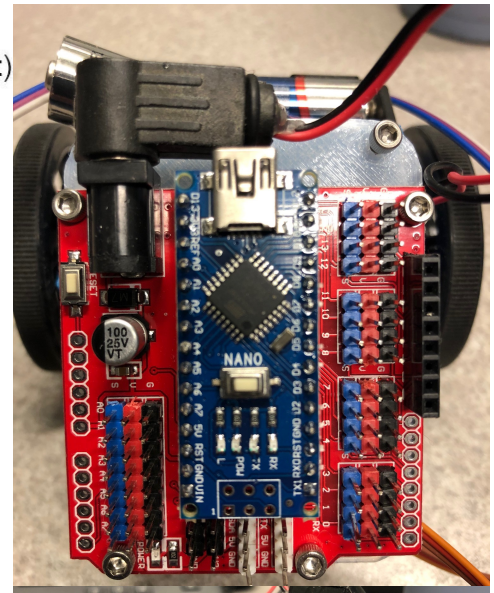
Dimention: Approx. 45x17x19mm

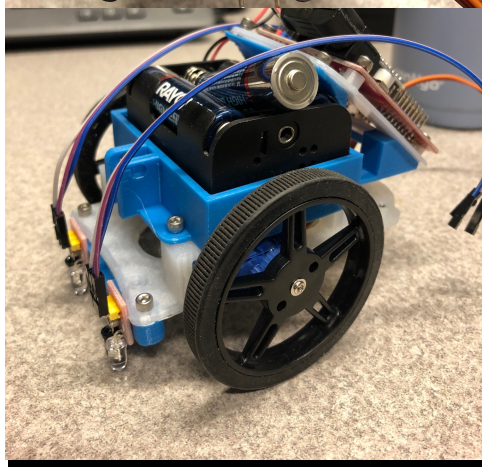
Cable Length: 245mm

Color: Blue(as the picture shows)

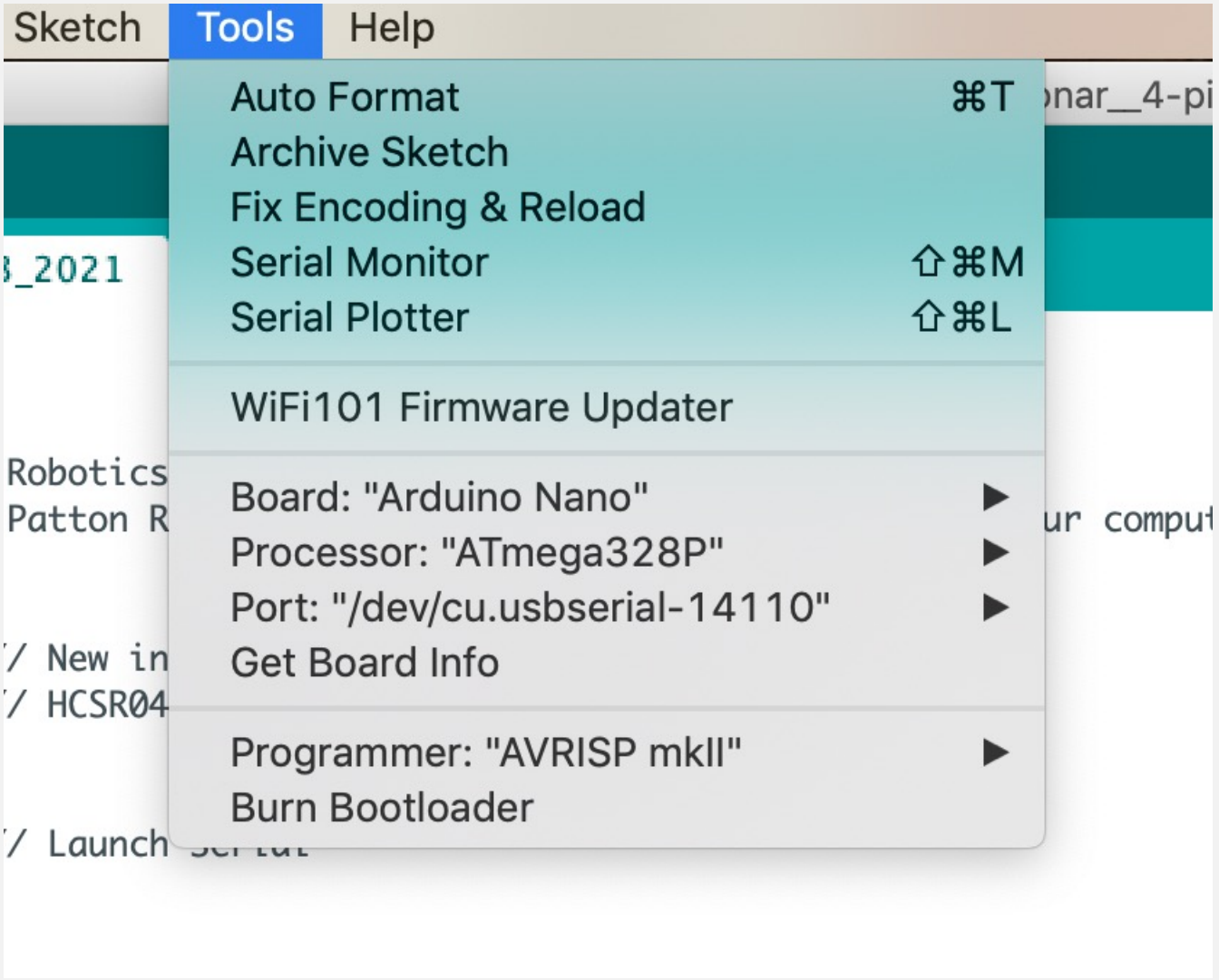
https://www.makerfocus.com/products/2pcs-nano-v3-0-atmega328p-microcontroller-board-for-arduino

The borads come assembled, you don't need to solder it by yourself.





ARDUINO IDE SET-UP



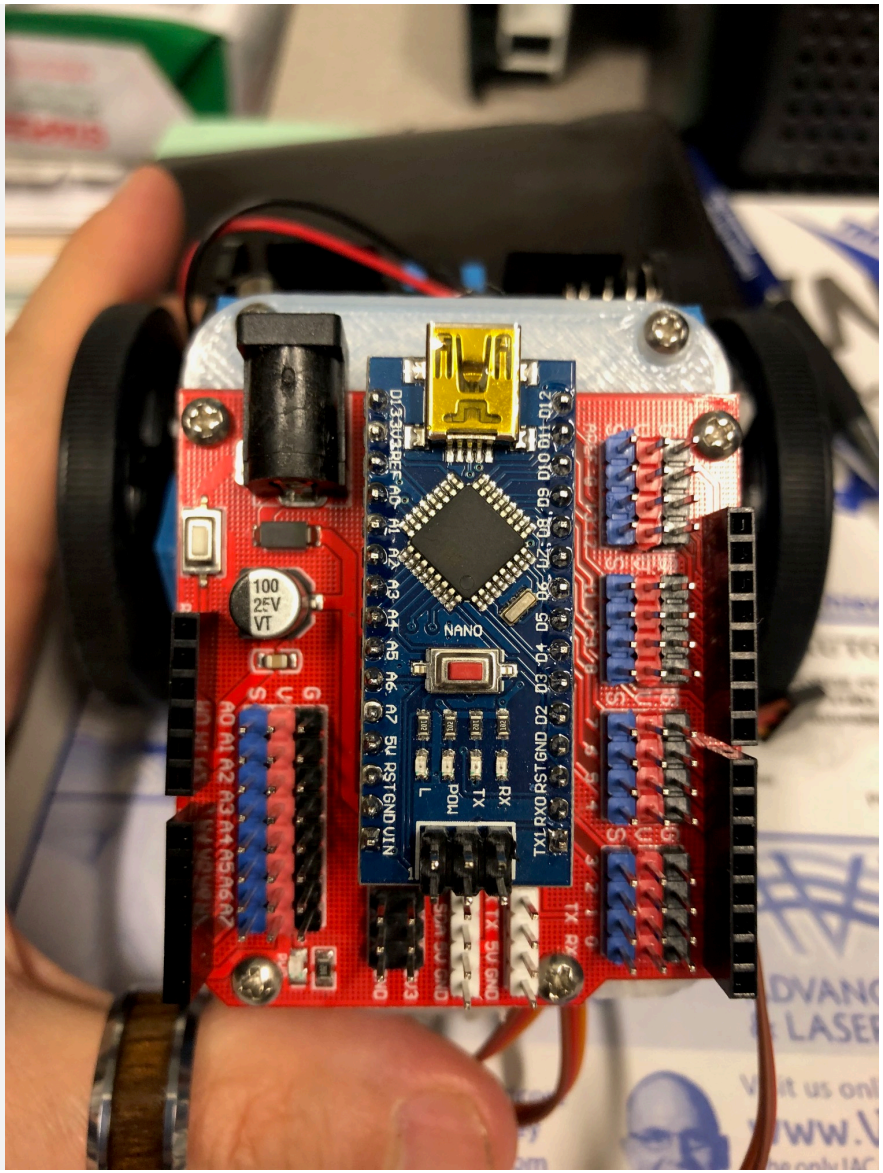
WIRING TO YOUR MICROCONTROLLER (USING THE CARRIER BOARD)

- Sensors are 3 or 4-pin.
  - 3-pin sensors have

    +V ---- connects to the +5V (red row of pins on the carrier board)

    G (Gnd) ---- connects to the G (black row of pins on the carrier board)

    S (Signal) ---- connects to the S (blue row of pins on the carrier board)
  - 4-pin sensors like your sonar have an extra signal pin
- A0, A1, A2.. A7 are for analog input devices (8)
- 2-13 are for digital inputs and outputs (12)
- 0 & 1 are for transmit/receive serial devices like LCD screens



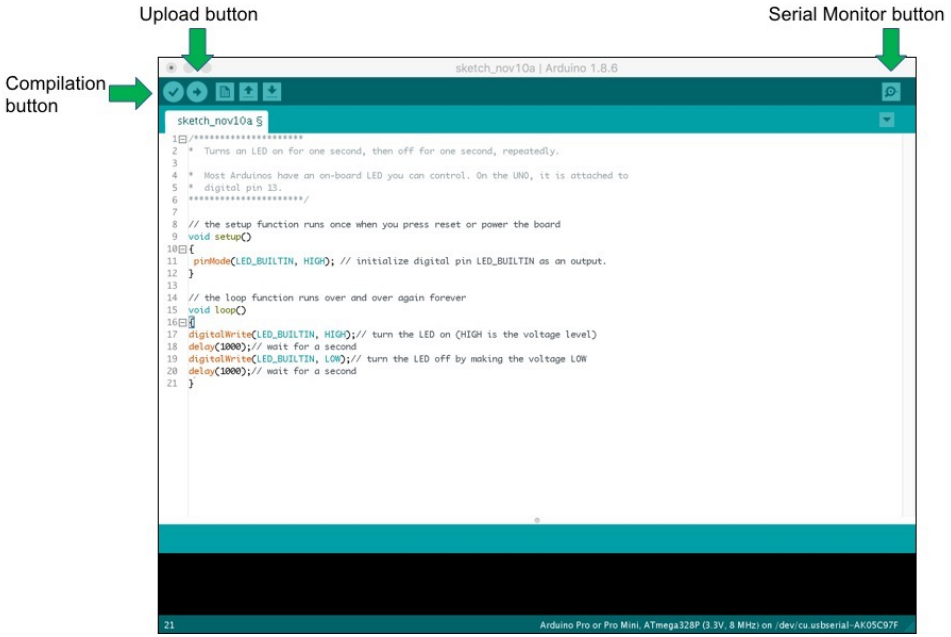
https://www.youtube.com/watch?v=nFbWXuR_2Ow

WHAT IS AN IDE?

- IDE = Integrated Development Environment
  - An integrated development environment (IDE) is a software suite that consolidates the basic tools developers need to write and test software.
  - Typically, an IDE contains a code editor, a compiler or interpreter and a debugger that the developer accesses through a single graphical user interface (GUI).
  - An IDE may be a standalone application, or it may be included as part of one or more existing and compatible applications.

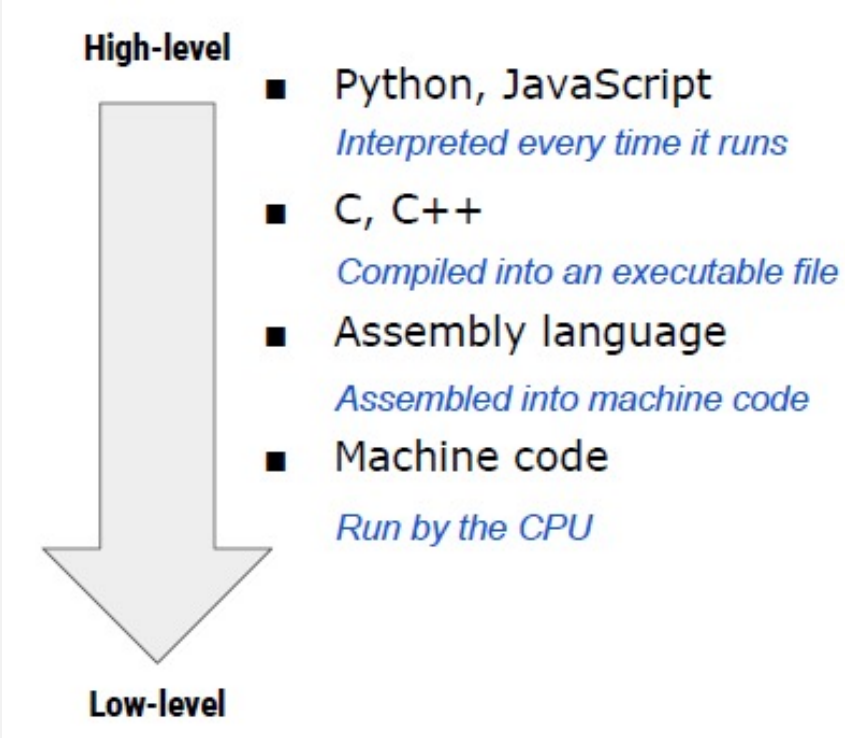
    http://searchsoftwarequality.techtarget.co/definition/integrated-development-environment

- JAVA uses Eclipse as its IDE
- We use the Arduino IDE to Program the Arduino with C++

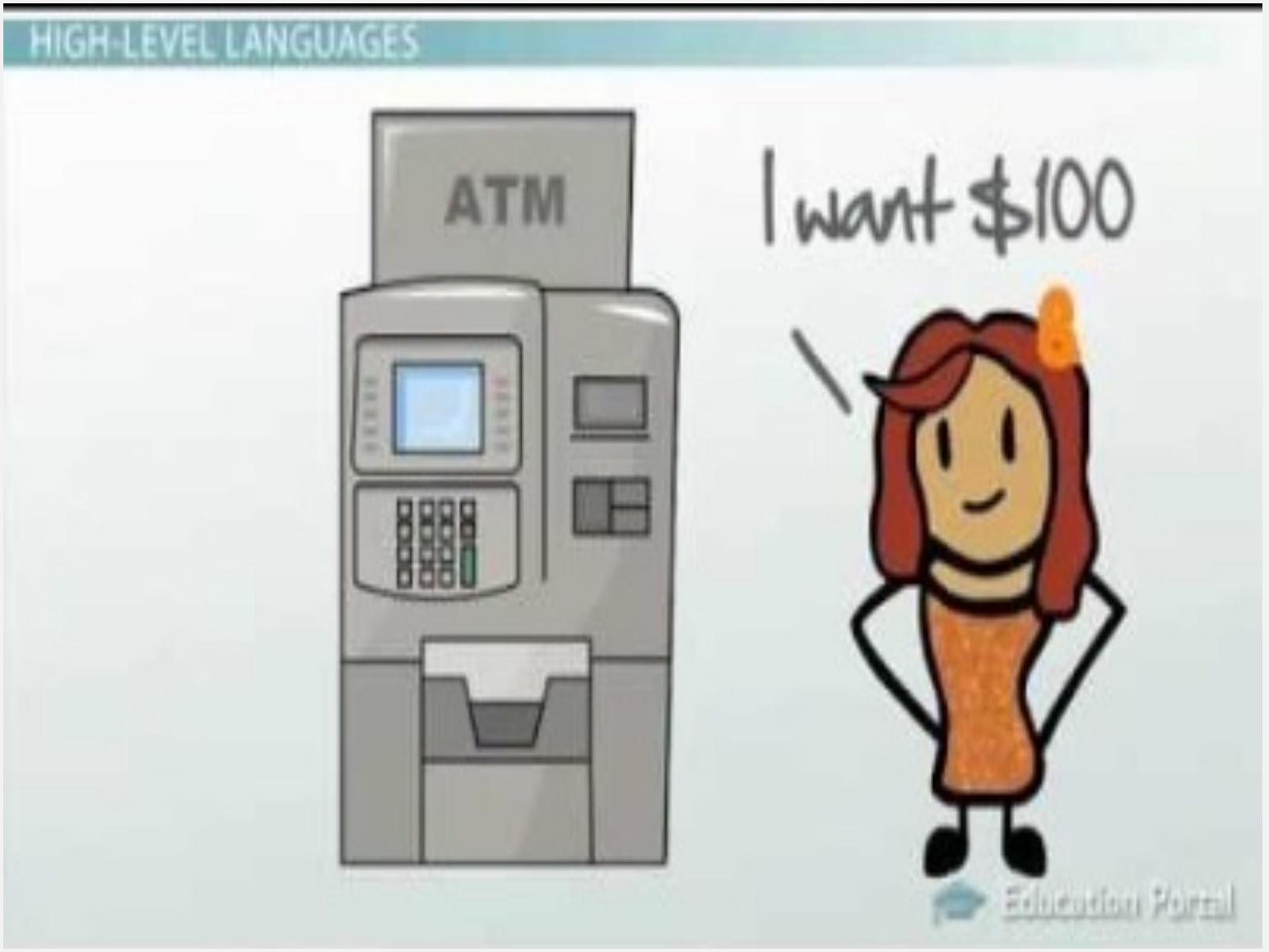


https://diy.waziup.io/sensors/introduction_Arduino_IDE/intro_Arduino_IDE.html

WHAT IS A HIGH-LEVEL LANGUAGE?



https://www.mrdfinch.com/high-and-low-level-languages.html



https://www.mrdfinch.com/high-and-low-level-languages.html

CODE FUNDAMENTALS (C++ SKETCH FORMAT / ORGANIZATION)

- Initialization Section (top)
- Setup Section (middle)
- Main Program Section (bottom)

```
int countUp = 0;                    //creates a variable integer called 'countUp'

void setup() {
Serial.begin(9600);                 // use the serial port to print the number
}

void loop() {
countUp++;                          //Adds 1 to the countUp int on every loop
Serial.println(countUp);            // prints out the current state of countUp
delay(1000);
}
```

https://www.arduino.cc/reference/en/language/variables/data-types/int/

CODE FUNDAMENTALS
(COMMON VARIABLE TYPES)

- Char - A data type used to store a character value. Character literals are written in single quotes, like this: 'A' (for multiple characters - strings - use double quotes: "ABC")
- Byte - A byte stores an 8-bit unsigned number, from 0 to 255
- Int - Integers are your primary data-type for number storage. On the Arduino Uno (and other ATmega based boards) an int stores a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767

https://www.arduino.cc/en/Reference/VariableDeclaration

CODE FUNDAMENTALS
(OTHER VARIABLE TYPES)

- Unsigned int - On the Uno and other ATMEGA based boards, unsigned ints (unsigned integers) are the same as ints in that they store a 2 byte value. Instead of storing negative numbers however they only store positive values, yielding a useful range of 0 to 65,535
- Long - Long variables are extended size variables for number storage, and store 32 bits (4 bytes), from -2,147,483,648 to 2,147,483,647
- Float - Datatype for floating-point numbers, a number that has a decimal point. Floating-point numbers are often used to approximate analog and continuous values because they have greater resolution than integers. Floating-point numbers can be as large as 3.4028235E+38 and as low as -3.4028235E+38. They are stored as 32 bits (4 bytes) of information
- Double - Double precision floating point number. On the Uno and other ATMEGA based boards, this occupies 4 bytes. That is, the double implementation is exactly the same as the float, with no gain in precision. On the Arduino Due, doubles have 8-byte (64 bit) precision.

https://www.arduino.cc/en/Reference/VariableDeclaration

CODE FUNDAMENTALS
(HOW TO DECLARE A VARIABLE)

- How to Declare a Variable (do this at the top of your code - global):

```
int var = val
```

where int = integer

var = variable

val = what you are assigning to the variable (initial number or pin)

- Global vs. Local
  - Global – top of the code
  - Local – inside a function

EXAMPLE C++ CODE

```
int countUp = 0;                // creates a variable integer called 'countUp'

void setup() {
Serial.begin(9600);             // use the serial port to print the number
}

void loop() {
countUp++;                      // adds 1 to the countUp int on every loop
Serial.println(countUp);        // prints out the current state of countUp
delay(1000);
}
```

CODE FUNDAMENTALS

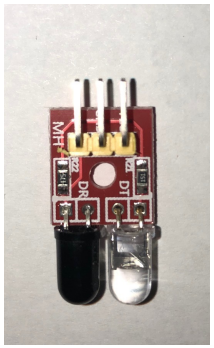
(ADDING IR SENSORS)

```
Left =  113     Right = 167
Left =  114     Right = 166
Left =  115     Right = 166
Left =  115     Right = 165
Left =  114     Right = 166
Left =  111     Right = 164
Left =  111     Right = 163
Left =  110     Right = 163
Left =  110     Right = 164
Left =  110     Right = 162
Left =  112     Right = 162
Left =  109     Right = 161
Left =  109     Right = 162
Left =  108     Right = 161
Left =  109     Right = 160
```

Arduino File Edit Sketch Tools Help

Front_Sensor_Basic_Test_Sumobot_6_02_2021 | A

Front_Sensor_Basic_Test_Sumobot_6_02_2021



```
//Front Sensor Basic Test - Sumobot
//Dr. John Wright
//6/2/2021

int sensor1 = A0;                                   //sensor1 is declared as an integer and connected to pin A0
int s1val = 0;                                      //s1val is declared as integer and set initially to zero
int sensor2 = A1;
int s2val = 0;

void setup() {
Serial.begin(9600);                                 //Establish serial buad rate
while(!Serial);                                     //Wait until good serial connection is established
}

void loop() {
s1val = analogRead(sensor1);                        //Read value from sensor1
s2val = analogRead(sensor2);
Serial.println((String)"Left =  "+ s1val + "     Right = "+ s2val); //Print a string to the monitor
delay(200);                                         //Wait 200ms
}
```

CODE FUNDAMENTALS (SETTING THE DIRECTION OF THE I/O)

- If having trouble, Use PinMode Command to set the direction of the I/O in the Void setup() Function
- **Arduino** (Atmega) pins **default** to **inputs**, so they don't need to be explicitly declared as **inputs** with **pinMode**() when you're using them as **inputs**. Pins configured this way are said to be in a high-impedance state. https://www.arduino.cc/en/Tutorial/Foundations/DigitalPins
- If using a library to control an output, PinMode may not be necessary

The code makes the digital pin 13 `OUTPUT` and Toggles it `HIGH` and `LOW`

```
void setup() {
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

CODE FUNDAMENTALS (BLINKING YOUR ON-BOARD LED)

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

CODE FUNDAMENTALS (LIBRARIES)

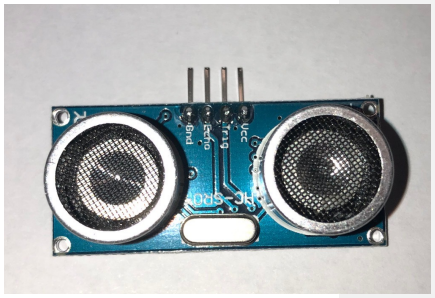
- The Arduino environment can be extended through the use of libraries, just like most programming platforms.
- Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from Sketch > Import Library.
- A number of libraries come installed with the IDE, but you can also download or create your own.

https://www.arduino.cc/en/reference/libraries

CODE FUNDAMENTALS (ADDING A SONAR SENSOR)

Sonar__4-pin__Test_Sumbot_6_3_2021

```
#include <HCSR04.h>
//Code & Library from Patton Robotics
//must get library file from Patton Robotics and install - point to folder on your computer
//Sketch, Import Library

HCSR04 Echo1(11,10);          // New instance of the class, use digital pins
                              // HCSR04(int EchoPin,int TrigPin)

void setup() {
  Serial.begin(9600);         // Launch Serial
}

void loop() {
  Echo1.ReadEchoCM();         // Get Data in Centimeters
  delay(10);                  // Give a chance to establish a new low on the trigger
                              // delay likely not needed, I just play it safe
Echo1.ReadEchoInches();       // Get Data in Inches
 Serial.print("CM = ");
 Serial.println(Echo1.CMs);
 Serial.print("Inches = ");
 Serial.println(Echo1.Inches);
 Serial.println(" ");
 delay(500);
}
```



```
CM = 14
Inches = 5

CM = 13
Inches = 5

CM = 14
Inches = 5

CM = 14
Inches = 5

CM = 13
Inches = 5
```

http://pattonrobotics.com/products/ultrasonic-sensor-and-cables

1) Download and install library onto computer Library Folder under Arduino first –-- drop the downloaded folder there

2) Link Library in Arduino
   SKETCH, IMPORT LIBRARY

CODE FUNDAMENTALS
(LIBRARIES FOR MOTOR CONTROL)

Servo_Library_Test_Code_Sumobot_6_2_2021

```
/*
Servo Library Test Code - Sumobot
Dr. John Wright
6/2/2021
*/

#include <Servo.h>
Servo leftservo;                        // create servo object to control our left servo
int spd = 110;                          // variable to store the servo speed 0 = full reverse, 180 is full forward, ~90 is stop

void setup() {
    leftservo.attach(2);                // leftservo connected to pin 2
}

void loop() {

    leftservo.write(spd);               // tell servo to go to position in variable 'pos'
    delay(20);                          // wait 20ms for the servo off-time to protect servo

}
```

CODE FUNDAMENTALS (IF STATEMENTS – PULLING IT ALL TOGETHER)

If_Statement_Test_Sumobot_6_02_2021 §

```
//Front Sensor Basic Test - Sumobot
//Dr. John Wright
//6/2/2021

#include <Servo.h>
Servo leftservo;                        // Create servo object to control our left servo
int spd1 = 150;                         // Variable to store the servo speed 0 = full reverse, 180 is full forward, ~90 is stop
int spd2 = 93;                          // Variable to store the servo speed 0 = full reverse, 180 is full forward, ~90 is stop

int sensor1 = A0;                       // sensor1 is declared as an integer and connected to pin A0
int s1val = 0;                          // s1val is declared as integer and set initially to zero

void setup() {
  Serial.begin(9600);                   // Establish serial buad rate
  while(!Serial);                       // Wait until good serial connection is established
  leftservo.attach(2);                  // leftservo connected to pin 2
}

void loop() {
  s1val = analogRead(sensor1);          // Read value from sensor1
  Serial.println(s1val);                // Print s1val to the monitor
  delay(200);                           // Wait 200ms
  if (s1val < 300) {
     leftservo.write(spd1);             // Tell servo to go spd1
     delay(20);                         // Wait 20ms for the servo off-time to protect servo
    }
  else {
  leftservo.write(spd2);
  delay(20);
  }
}
```

VIDEO

CODE FUNDAMENTALS (FUNCTIONS)

- A function is simply a subroutine.
  - Segmenting code into functions allows a programmer to create modular pieces of code that perform a defined task and then return to the area of code from which the function was "called". The typical case for creating a function is when one needs to perform the same action multiple times in a program.
  - For programmers accustomed to using BASIC, functions in Arduino provide (and extend) the utility of using subroutines (GOSUB in BASIC).
  - There are two required functions in an Arduino sketch, setup() and loop(). Other functions must be created outside the brackets of those two functions.

https://www.arduino.cc/en/Reference/FunctionDeclaration

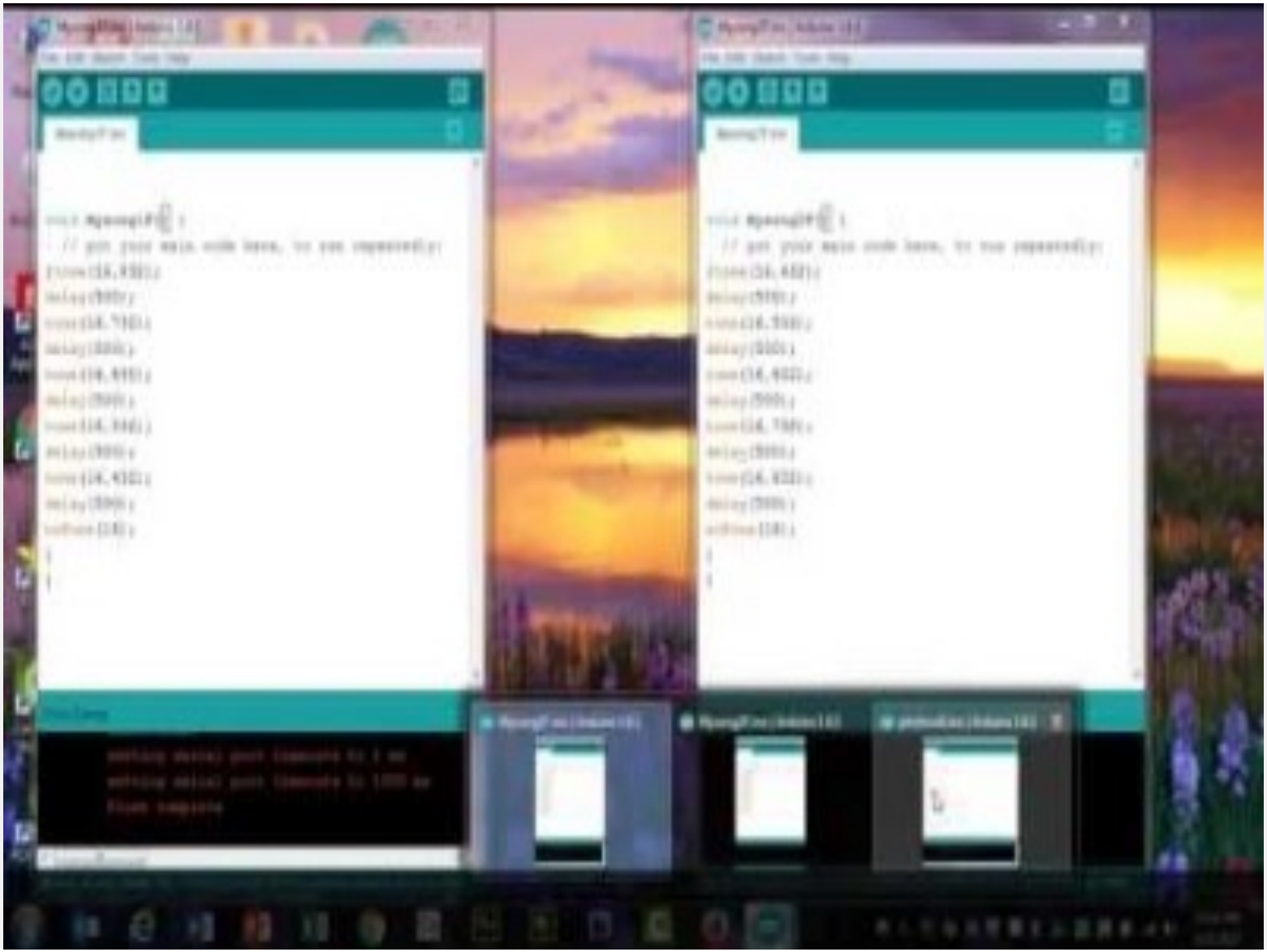
CODE FUNDAMENTALS (FUNCTIONS)

- Standardizing code fragments into functions has several advantages:
  - Functions help the programmer stay organized. Often this helps to conceptualize the program.
  - Functions codify one action in one place so that the function only has to be thought out and debugged once.
  - This also reduces chances for errors in modification, if the code needs to be changed.
  - Functions make the whole sketch smaller and more compact because sections of code are reused many times.
  - They make it easier to reuse code in other programs by making it more modular, and as a nice side effect, using functions also often makes the code more readable.

https://www.arduino.cc/en/Reference/FunctionDeclaration

https://www.youtube.com/watch?v=DrDB4oYtJrA&t=1s

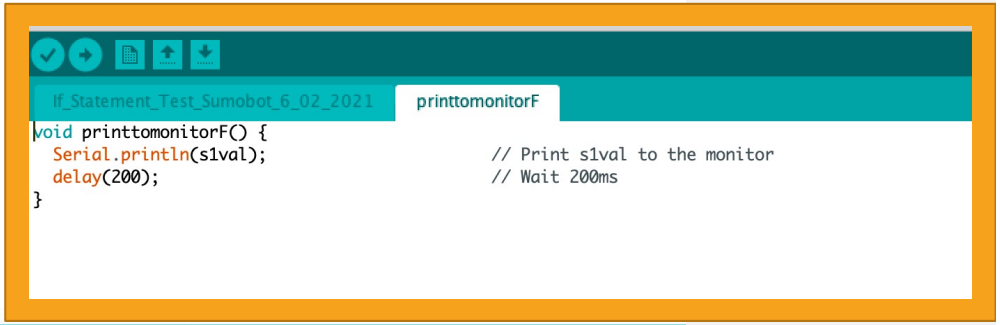
```
void loop() {
  sensorread();                          // Call sensorread subroutine / function
  printtomonitor();                      // Call printtomonitor subroutine / function
  if (s1val < 300) {
     leftservo.write(spd1);              // Tell servo to go spd1
     delay(20);                          // Wait 20ms for the servo off-time to protect servo
    }
  else {
  leftservo.write(spd2);
  delay(20);
  }
}

void sensorread() {
  s1val = analogRead(sensor1);           // Read value from sensor1
  //add more sensors here
}
void printtomonitor() {
  Serial.println(s1val);                 // Print s1val to the monitor
  delay(200);                            // Wait 200ms
}
```



https://www.youtube.com/watch?v=SCa_QRimtLI

If_Statement_Test_Sumobot_6_02_2021 | printtomonitorF

```
//Front Sensor Basic Test - Sumobot
//Dr. John Wright
//6/2/2021

#include <Servo.h>
Servo leftservo;                        // Create servo object to control our left servo
int spd1 = 150;                         // Variable to store the servo speed 0 = full reverse, 180 is full forward, ~90 is stop
int spd2 = 93;                          // Variable to store the servo speed 0 = full reverse, 180 is full forward, ~90 is stop

int sensor1 = A0;                       // sensor1 is declared as an integer and connected to pin A0
int s1val = 0;                          // s1val is declared as integer and set initially to zero

void printtomonitorF();

void setup() {
  Serial.begin(9600);                   // Establish serial buad rate
  while(!Serial);                       // Wait until good serial connection is established
  leftservo.attach(2);                  // leftservo connected to pin 2
}

void loop() {
  sensorread();                         // Call sensorread subroutine / function
  printtomonitorF();                     // Call printtomonitor subroutine / function
  if (s1val < 300) {
     leftservo.write(spd1);             // Tell servo to go spd1
     delay(20);                         // Wait 20ms for the servo off-time to protect servo
    }
  else {
  leftservo.write(spd2);
  delay(20);
  }
}

void sensorread() {
  s1val = analogRead(sensor1);          // Read value from sensor1
  //add more sensors here
}
```

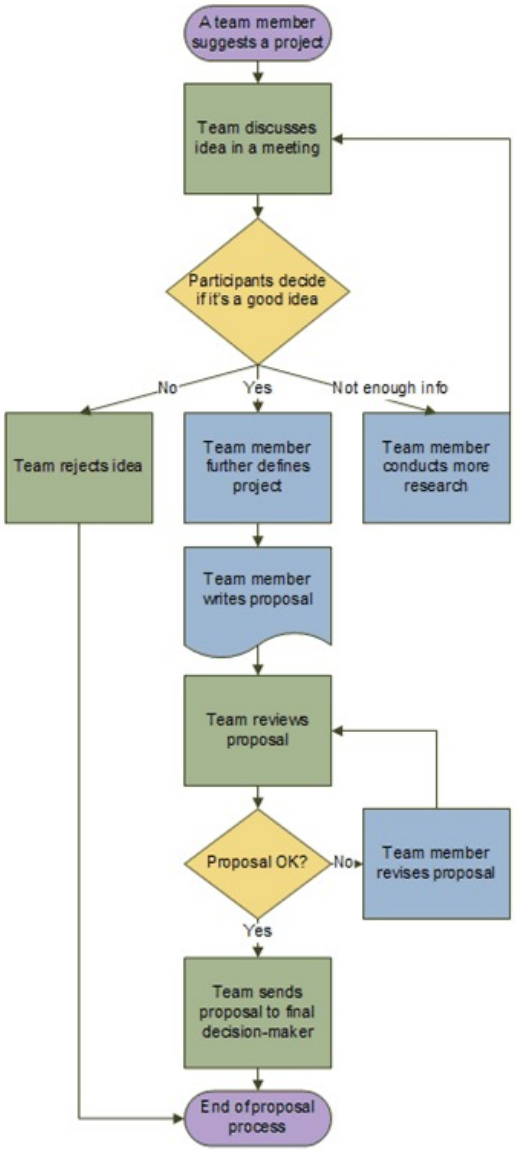


ALGORITHMS, PSEUDOCODE, AND CODE

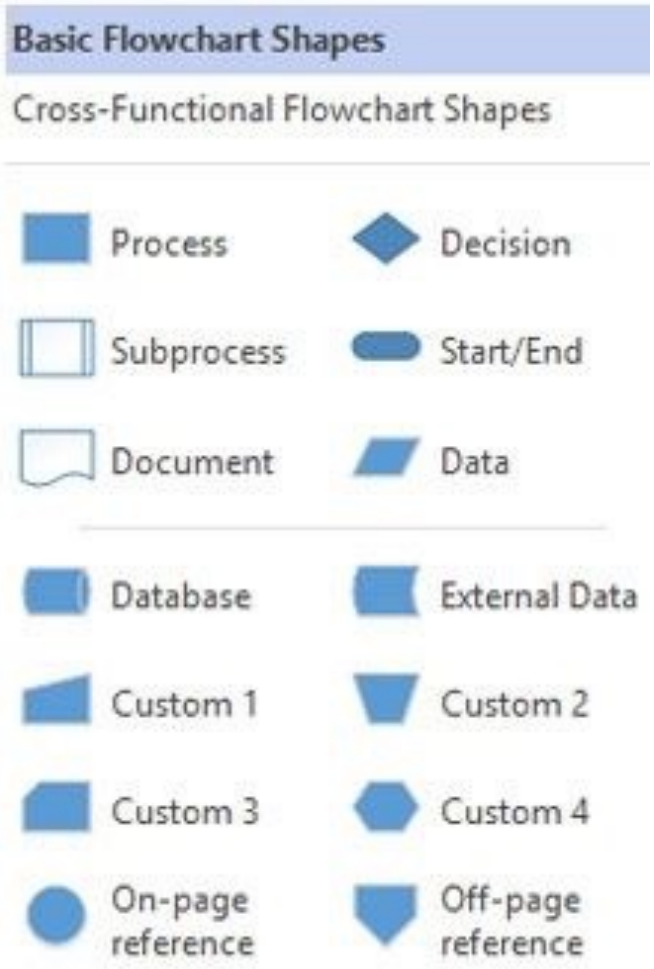
- An Algorithm is your plan/idea (how to solve a problem)
  - May be expressed in many different ways
  - Mathematical Expression
  - Pseudo Code (written text)
- Pseudocode is the written expression of the Algorithm
  - It is simply a description on how your program should work in plain English or another language
- Code – this is what you program (Syntax) to enact your algorithms
- Other notes:
  - Some people are great at code
  - Some are great at developing algorithms
    - Innovation really comes great algorithms!
    - Optimization comes from great code!

WHAT IS A FLOWCHART?

- Flowcharts allow one to see a pictorial representation of the process.
- They make it easier to understand the process at hand!
- MS Visio is a great tool for developing flowcharts as you can easily drag and drop the symbols.



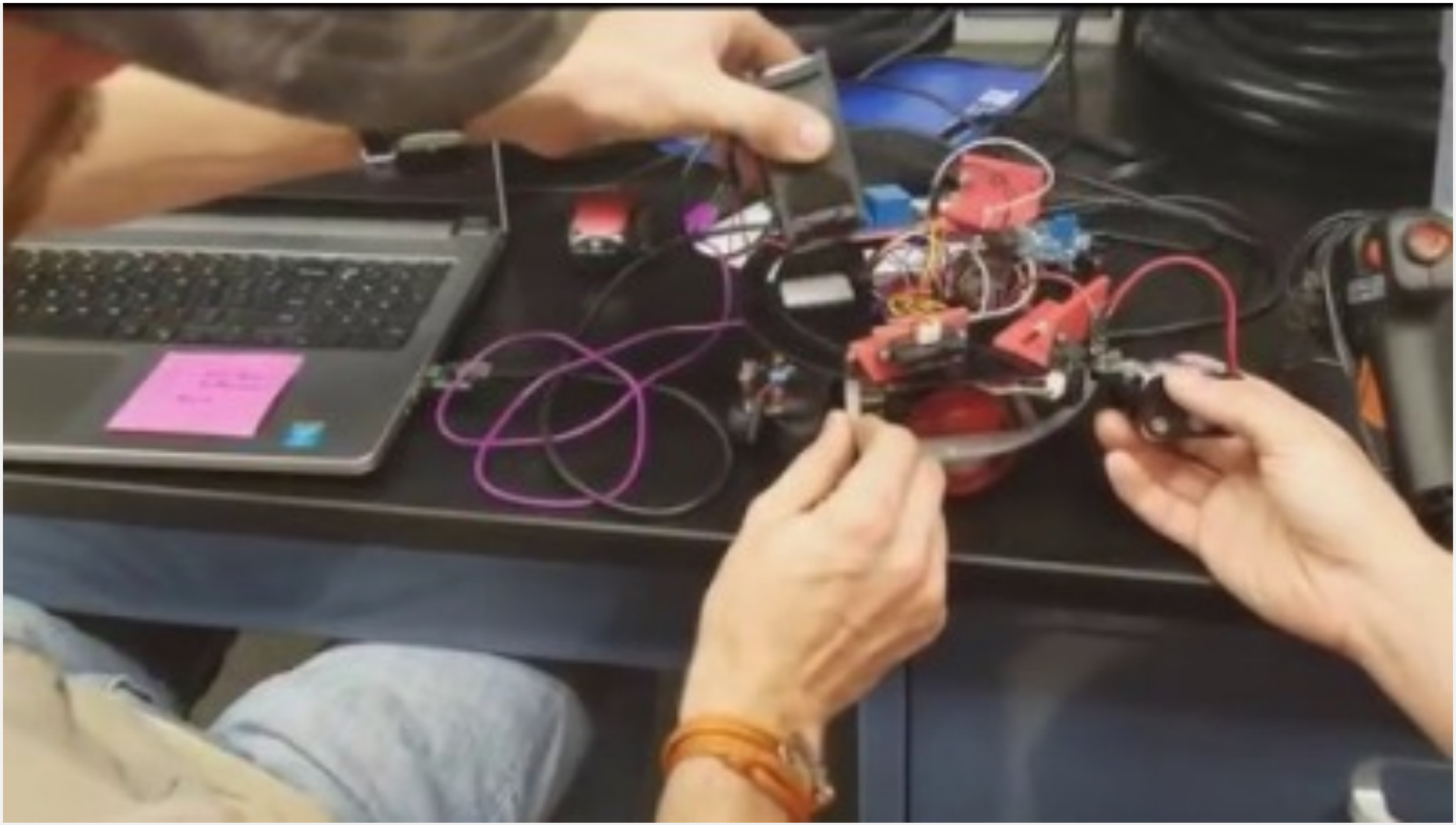
BASIC FLOWCHART SYMBOLS



https://www.youtube.com/watch?v=2rZY8iX8Mdw&t=2s

WHAT IS MEANT BY REAL-TIME I/O?

- Real-time I/O are programmed devices collect data and provide data or commands to other devices external to the computer.
- This is what separates a roboticist or controls engineer from a computer scientist.



https://www.youtube.com/watch?v=wSsmNSqUrsw

https://www.youtube.com/watch?v=z5_2xkOpVHU&t=2s