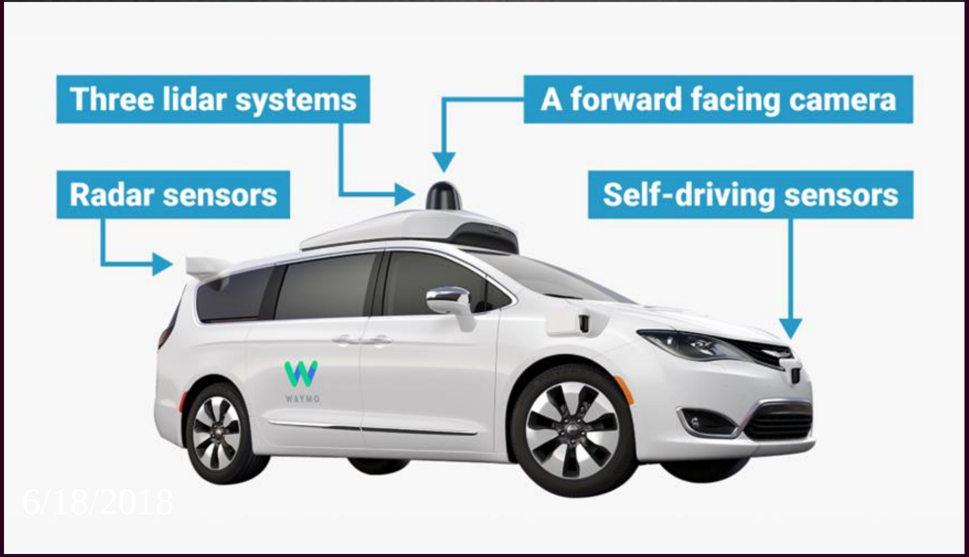
# Hello



- Autonomous car/ self driving vehicle
- Sensors – environment data as input
- Navigation without human intervention

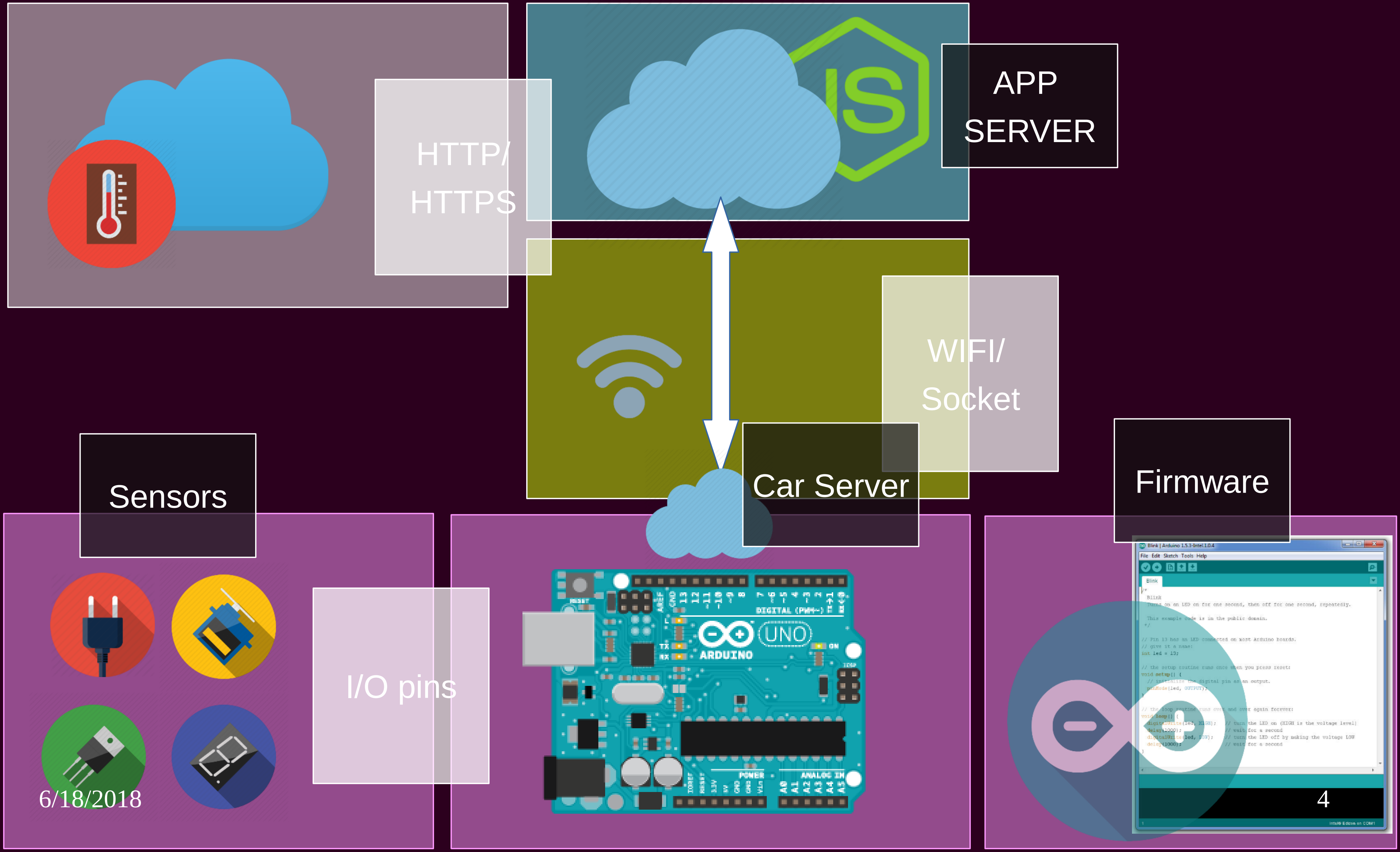
# Autonomous Wheelchair

- Sensing systems: Ultrasonic sensor, Radar, Lidar, Cameras, GPS, UWB etc.
- Communication Module: Wi-Fi, BLE, DSRC, etc.
- On board controller PC: computer for data processing
- Mechanical Design considerations of sensors placements etc.

# Autonomous Wheelchair

- Watch a video!

# System Block Diagram



APP SERVER

HTTP/ HTTPS

WIFI/ Socket

Car Server

Sensors

I/O pins

Firmware

```
/*
  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.
// 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
}
```

# Assembly!

- Duration: 1 hour

Microcontroller

Arduino

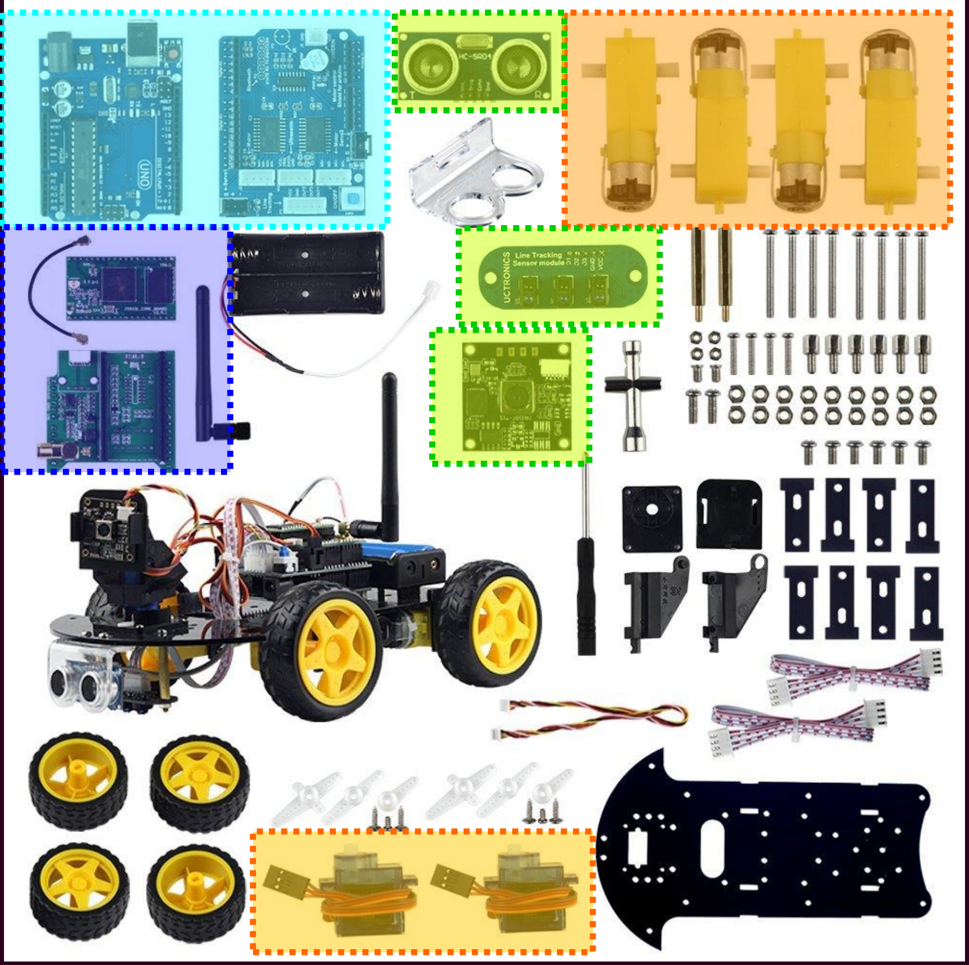
Comm module

Extension Shield

Mech. Parts

Wheels & mounts

Screws & Chassis



Sensors

Ultrasonic Sensor

IR Sensor

Camera

Actuators

DC Motors

Servos

# Notes

- Left and Right of motors
- Ground & Power line
- Batteries will be provided upon completion of assembly
- Duration: 1 hour

# IoT 101

- Content Overview Day 1& Day 2:
  - RobotCar assembly (done!)
  - Overview of Arduino
    - Hardware & Software
  - Sensor Nodes
  - Programming

# Arduino

- Open-source electronics prototyping platform
- Open-source software/ hardware
  - Hardware : Atmel 8-bit microcontroller Atmega8 Series
  - Standard programming language compiler,
  - Boot loader
- You can make full use of open source materials on the Internet!

# Arduino

D13 LED

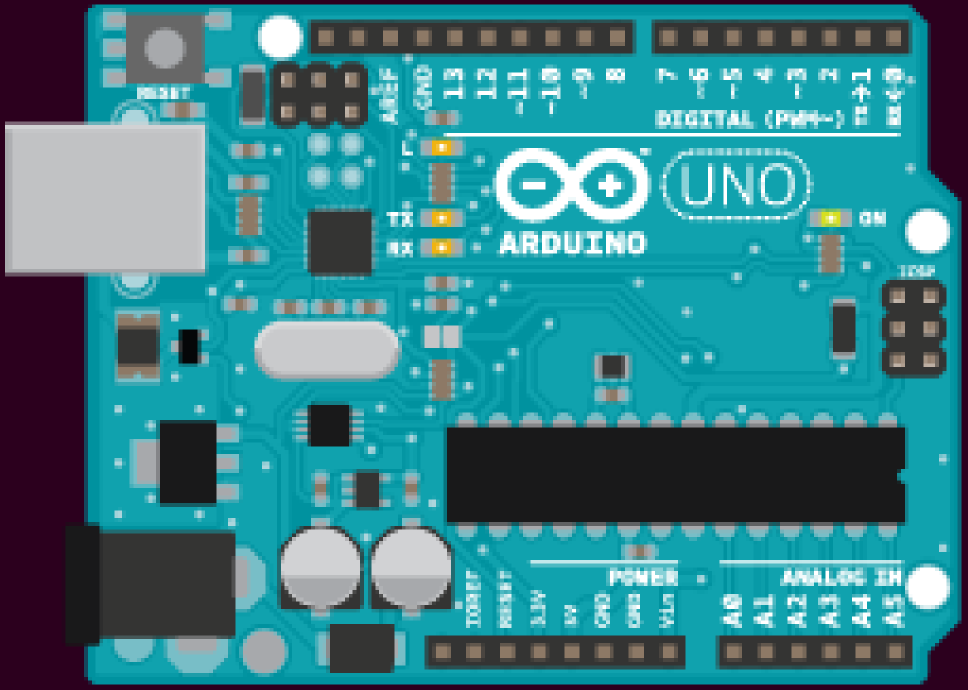
Tx/Rx LEDs

USB to Serial IC

USB Jack

5V Regulator

DC Power Jack



Digital I/O Pins

LED

Atmel Atmega 328

Analog I/O Pins

Power Pins

# Arduino IDE

- Integrated Development Environment (IDE)
  - **Programming tool** with built-in compiler
  - Compiler changes the code to machine language and loads into the Arduino
  - Checks for errors
- Has **libraries**
  - expand capabilities
- Programs are called **sketches**

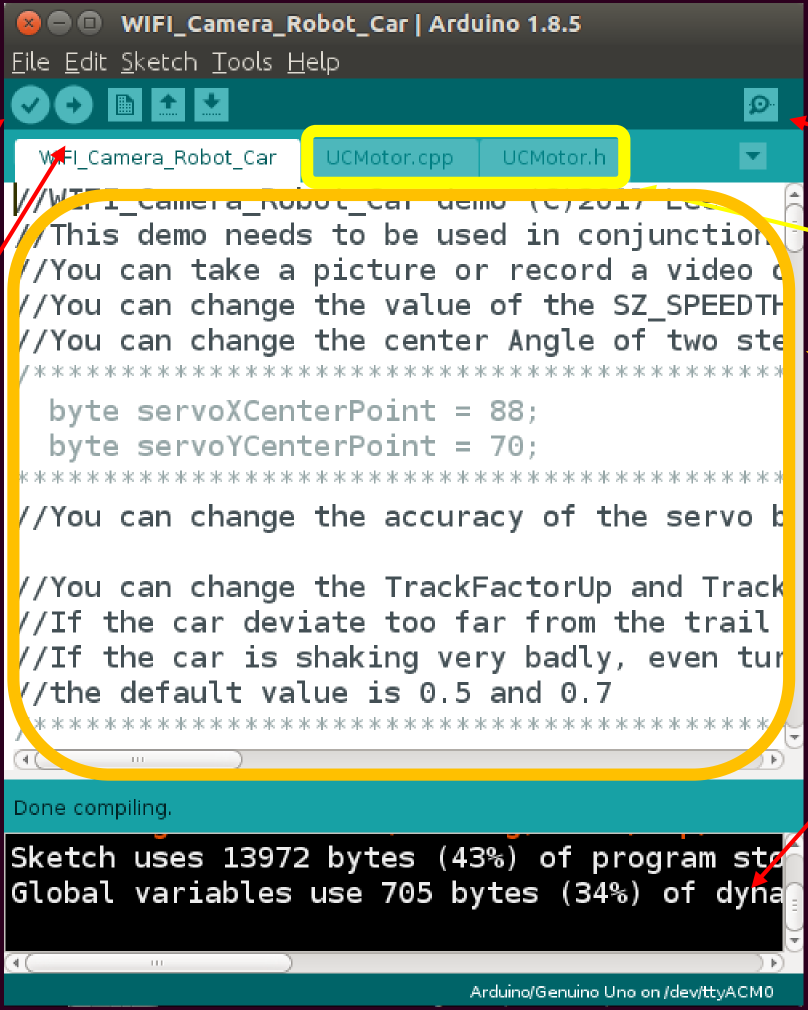
# Arduino IDE

- Sequence:
  - **Actions to be taken**: Write sketch
  - **Sanity Check**: Compile program into binary data
  - **Tell Arduino**: Upload bits through USB cable to Arduino
  - TX/RX blink

# Arduino IDE

Upload button when succeed, ‘done uploading’

Sketch button when succeed, ‘done compiling’


WIFI_Camera_Robot_Car | Arduino 1.8.5
File Edit Sketch Tools Help
WIFI_Camera_Robot_Car
UCMotor.cpp
UCMotor.h

```
//WIFI_Camera_Robot_Car demo (C)2017 Lee
//This demo needs to be used in conjunction
//You can take a picture or record a video c
//You can change the value of the SZ_SPEEDTH
//You can change the center Angle of two ste
/*****************************************
  byte servoXCenterPoint = 88;
  byte servoYCenterPoint = 70;
******************************************
//You can change the accuracy of the servo b

//You can change the TrackFactorUp and Track
//If the car deviate too far from the trail
//If the car is shaking very badly, even tur
//the default value is 0.5 and 0.7
/*****************************************
```

Done compiling.

```
Sketch uses 13972 bytes (43%) of program sto
Global variables use 705 bytes (34%) of dyna
```

Arduino/Genuino Uno on /dev/ttyACM0

Serial monitor button

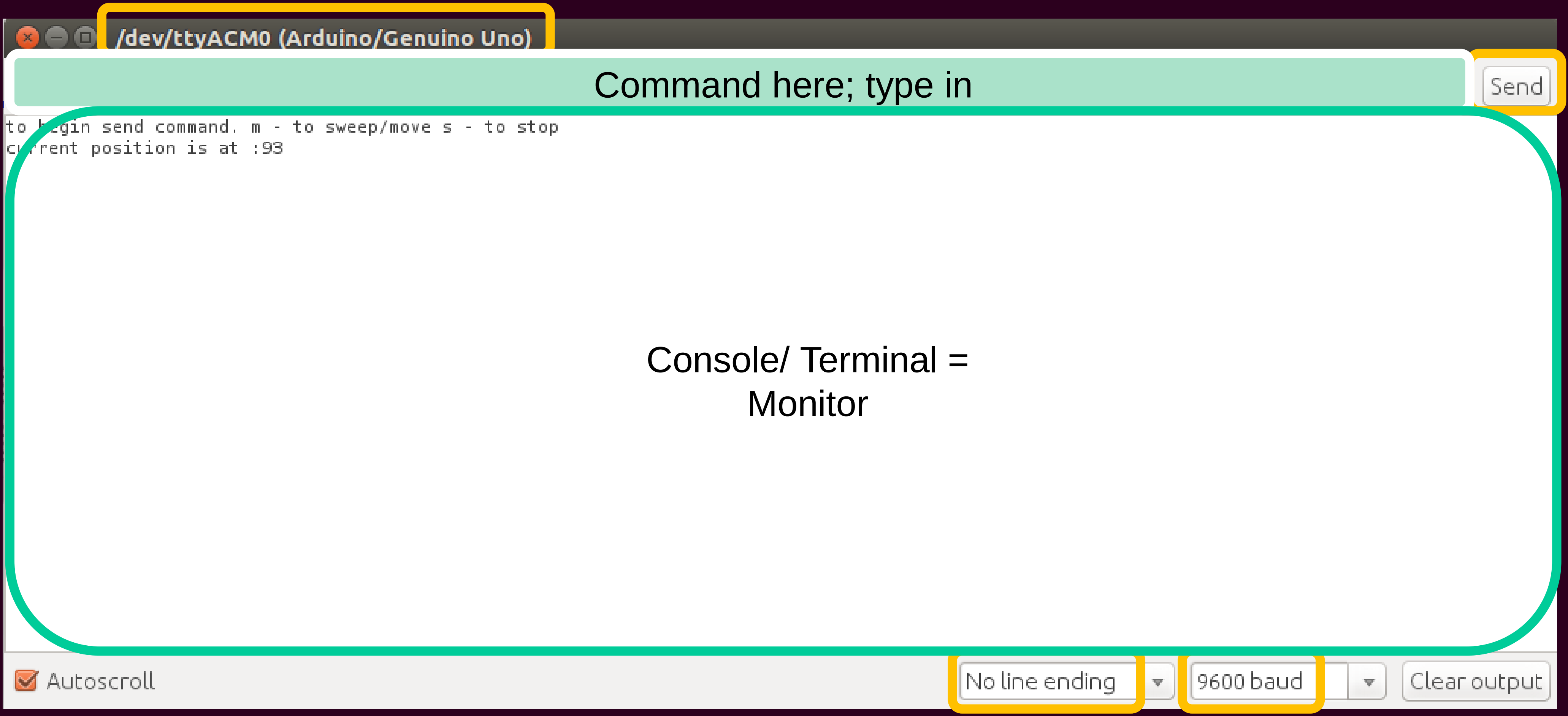
library

sketch

Status monitor/ debug console checks for any error messages

# Arduino IDE

Port Indication



/dev/ttyACM0 (Arduino/Genuino Uno)

Command here; type in

```
to begin send command. m - to sweep/move s - to stop
current position is at :93
```

Console/ Terminal = Monitor

☑ Autoscroll

No line ending

9600 baud

Configure how lines of commands are written

Baud Rate

# Hands-on Activity: Flash in

- **UCrobotics.ino (/sensor/RobotCar/example/WIFI_Camera_Robot_Car folder)**
- Try :
  - Get firmware from
    - https://github.com/UCTRONICS/WIFI_Camera_Smart_Robot_Car
    - Compile, and launch the firmware into Arduino

# Hands-on Activity: APK

- **RobotCar.apk file (/sensor/apk folder)**
- Try:
  - Get apk from:
    - https://github.com/UCTRONICS/WIFI_Camera_Smart_Robot_Car/tree/master/APP_Controller
  - Give permission for debugging mode
  - Transfer the apk to your phone
  - Install apk
  - Launch and start controlling the car!

# Try!

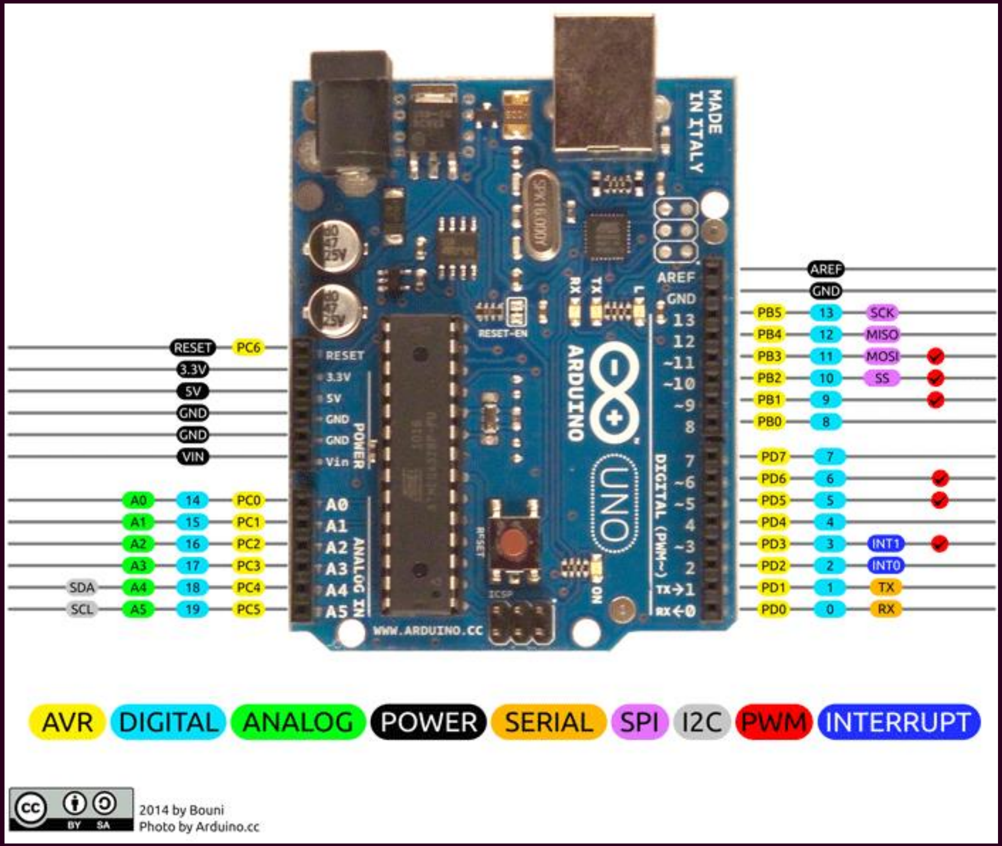
- Move the car around☺
- Capture images
- Data analytics module lesson later

# Arduino Sketch Structure

- Software Programming
- main parts
  - Structure
    - Setup(), loop(), functions()
  - Values (variables, constants)

# Input/output

- During the course, you will program microcontrollers to control actuators
  - How? Input/output concept – analog vs digital



2014 by Bouni
Photo by Arduino.cc

# Input/output

- There are **digital i/o** and **analog pins** on Arduino
  - pinMode(pin, mode)
  - Sets pin to either INPUT (read) or OUTPUT (control)
    - digitalRead(pin), digitalWrite(pin), AnalogRead(pin), AnalogWrite(pin)
  - Writes HIGH or LOW to a pin
    - e.g. LED ON or OFF
- Output pins can provide 40mA of current. Writing HIGH to an input pins installs a 2K ohm pull up

# Pins

- Digital pins:  3, 5, 6, 9, 10, 11
- Analog pins: A0 to A5
- 0V or 5 V  (digitalWrite) or output a PWM (analogWrites) signal
- Full range is 0 to 255
  - i.e. 100 = 39 % duty cycle
- AnalogRead values go from 0 to 1023

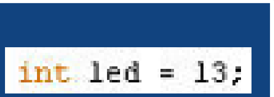
# Simple Handbook

**Sketch:**

Setup(): run once upon bootup

Loop(): runs continuously

**Data type:**



```
int led = 13;
```

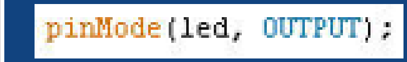
Int – number storage

int var = val

Var – int variable name

Val – value that you assign to that variable

**Pin Assignment:**



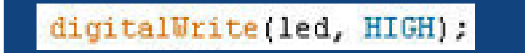
```
pinMode(led, OUTPUT);
```

pinMode(pin, mode)

Pin – number of the pin whose mode you wish to set

Mode: INPUT, OUTPUT

**Pin Control:**



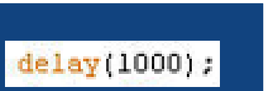
```
digitalWrite(led, HIGH);
```

digitalRead(pin, value)

digitalWrite(pin, value)

Value – HIGH(5V, 3.3V, etc) or LOW (0V)

**Delay(ms)**



```
delay(1000);
```

Pause (milliseconds i.e. 1000ms = 1 seconds)

**; (semi-colon)**

To indicate the end of line, it is a must!

# Hands-on Activity: Blink!

- **Blink.ino** (/sensor folder)
- Try:
  - a. Turn on the LED
  - b. Blink the LED
- Note:
  - int ledPin = 13; //built-in LED on Arduino

# Programming

- If/else
- if(condition){//statements}
- Checks for a condition and executes if true

- Comparison operators
  - X == Y
  - X != Y
  - X < Y
  - X > Y
  - X <= Y
  - X >= Y

# Programming

- For loop
- for(initialization; condition; increment) {//statements}
- To repeat enclosed statements until condition is met.

# Programming

- while
- while(condition) {//statements}
- To repeat enclosed statements continuously until (condition) is false.

# Programming Exercise

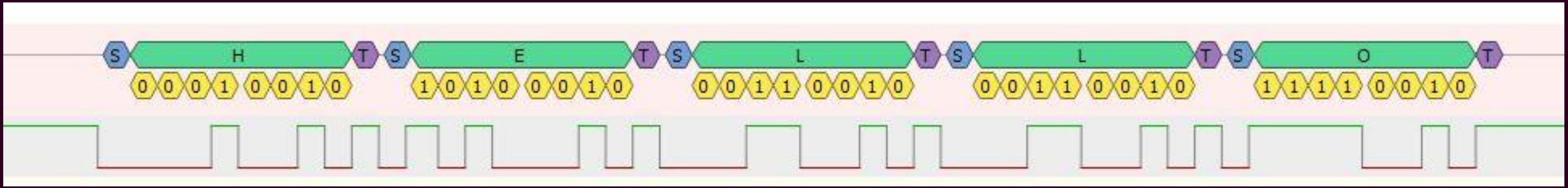
- **if_Loop.ino** && **forLoop.ino** && **while_loop** (/**sensor folder)**
- Try :
  - Run the code and get familiarized with the concept.

# Make Arduino Talk & Listen

- Now, we will make Arduino **communicate** with other devices, your computer, etc.
- Arduino has serial port (a.k.a UART)
  - TX (pin 1), RX(pin 0)
- Serial monitor ('tester' between computer and Arduino) to communicate with Arduino board
- Later, we will send command to Arduino's serial port to control leds, motors

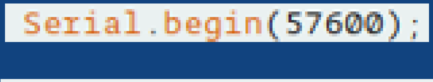
# Serial Communication

- Information is transmitted as zeros and ones – bits
- Serial as data is broken down into bits, each sent one after the other down a single wire
- Single ASCII character is sent as:
- Baud Rate (i.e. 300, 600, 4800, 9600, 19200 etc.)



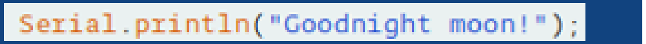
# Simple Handbook

1. initial setting for serial – set baud rate



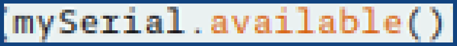
```
Serial.begin(57600);
```

2. println() to transmit data to the serial port



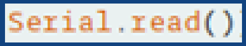
```
Serial.println("Goodnight moon!");
```

3. available() to wait for incoming serial data in the serial port



```
mySerial.available()
```

4. read() to grab incoming serial data received on RX pin

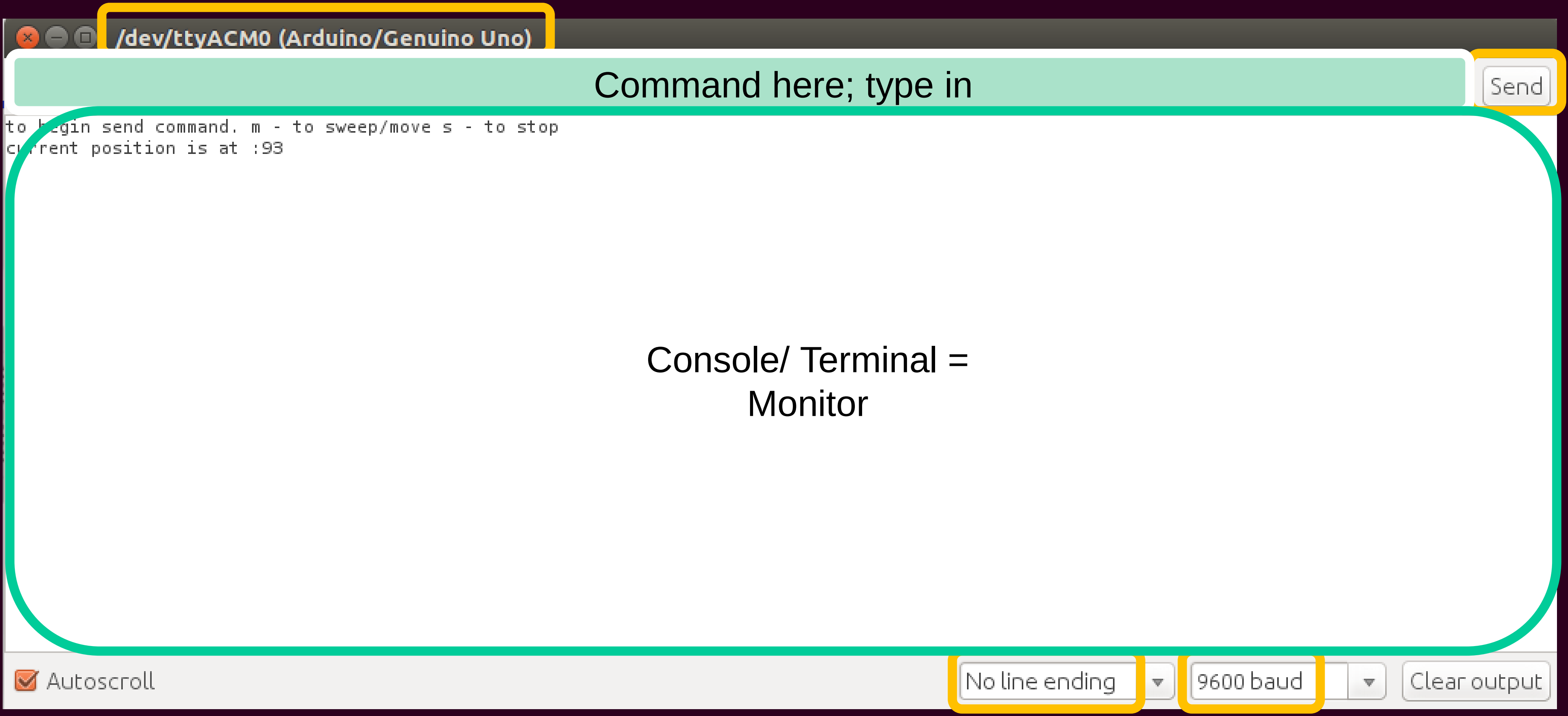


```
Serial.read()
```

- *“A” and ‘A’ are different.* “A”: string ‘A’: character (char)
- Char holds up one character. How to send more than one character command?
- End of line indication takes up bytes too(/r/n). Serial monitor setting is at no line ending.
- Println(“hello”) i.e. hello/r/n

# Arduino IDE

Port Indication



/dev/ttyACM0 (Arduino/Genuino Uno)

Command here; type in

```
to begin send command. m - to sweep/move s - to stop
current position is at :93
```

Console/ Terminal = Monitor

☑ Autoscroll

No line ending

9600 baud

Configure how lines of commands are written

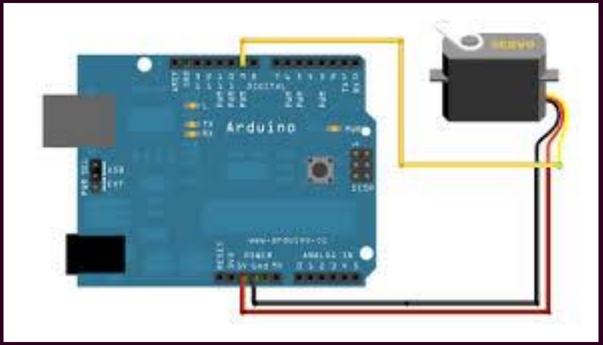
Baud Rate

# Hands-on Activity: Chatty Arduino

- **ChattySerial.ino** (/sensor folder)
- Try:
  - Compile, flash, open serial monitor to explore
  - Turn on/ off LED with serial command
  - ‘on’ -> turn on the LED
  - ‘off’ -> turn off the LED
  - Play with baud rate in the sketch and serial monitor to see the difference

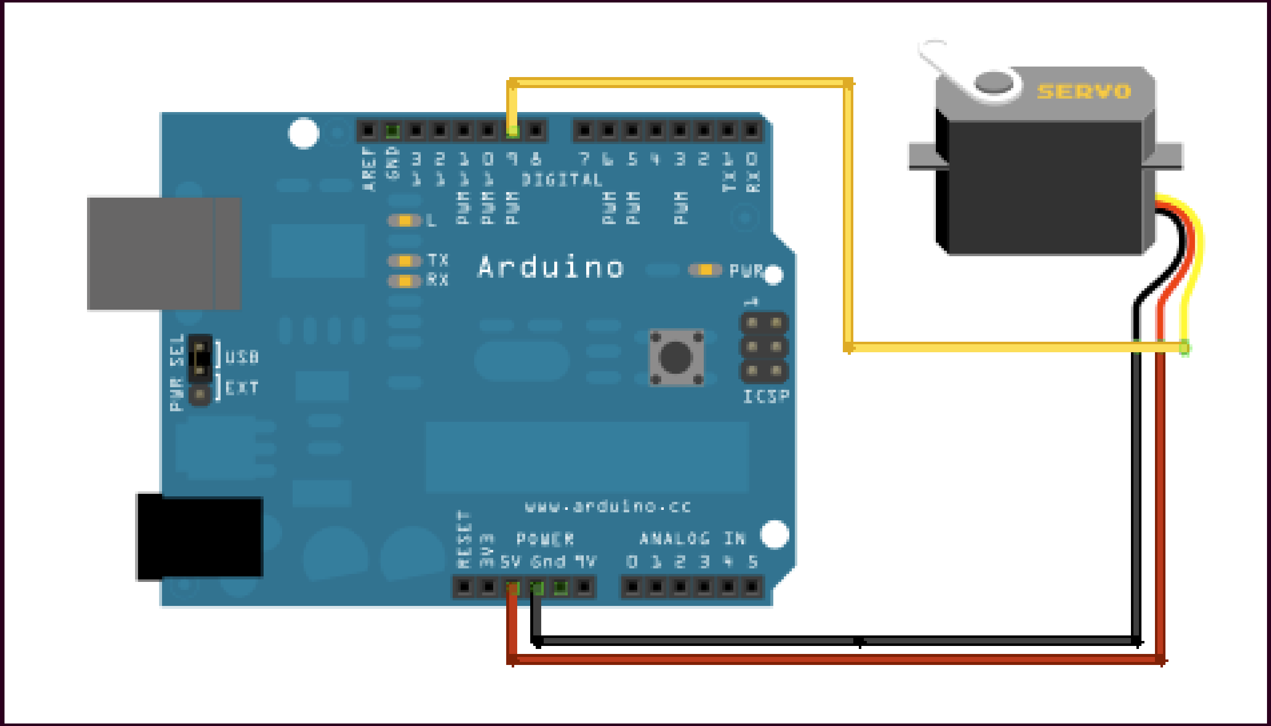
# Servo

- Rotary actuator with built-in feedback mechanism
- Precise control of angular position, velocity, and acceleration
- 1 of 2 types: Continuous rotation
  - Pulse control (PWM) to control Direction and Speed
- 2 of 2 types: ~180 degree rotation
  - To position
- Small DC motors
- Gearbox with small plastic gears to reduce RPM and increase output torque
- Special electronics to interpret a pulse signal and deliver power to the motor



# Looking at servo hardware

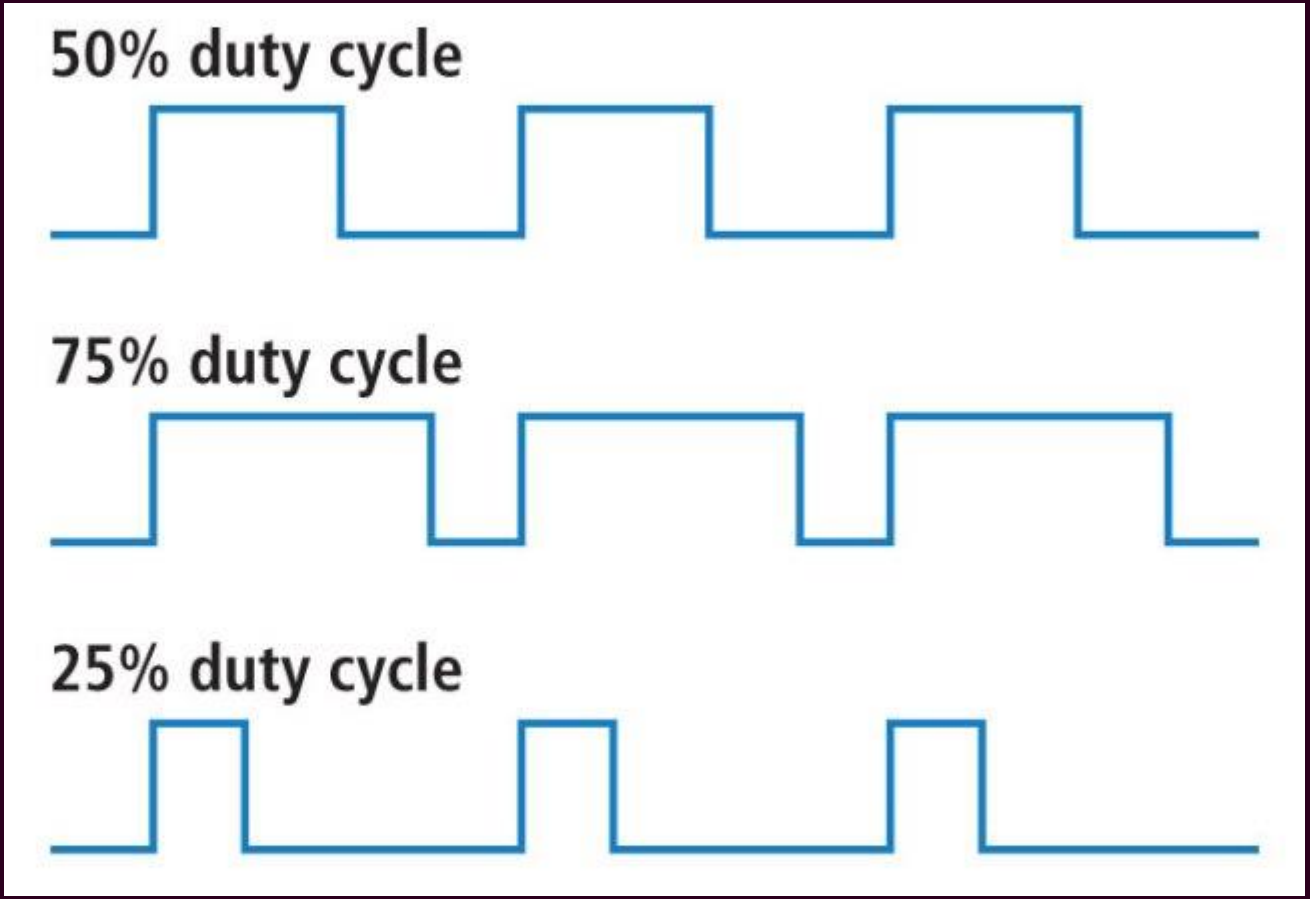
- Three wires
  - Ground, power, control signal
- Control signal results in moving the shaft to an angular position



# Control signal

- Pulse train
- PWM is used for the control signal of servo motors
  - Duration of the positive pulse sets position of the servo shaft (not speed)
- Typical pulse frequency is at 20ms
- Pulse width determines position
  - Typically at 1ms to 2ms

# PWM



# Servo library

- Three components of the Servo Library
  - Create the servo object

```
Servo my_servo_object;
```

Name of the object is like a variable name.

  - Attach the object

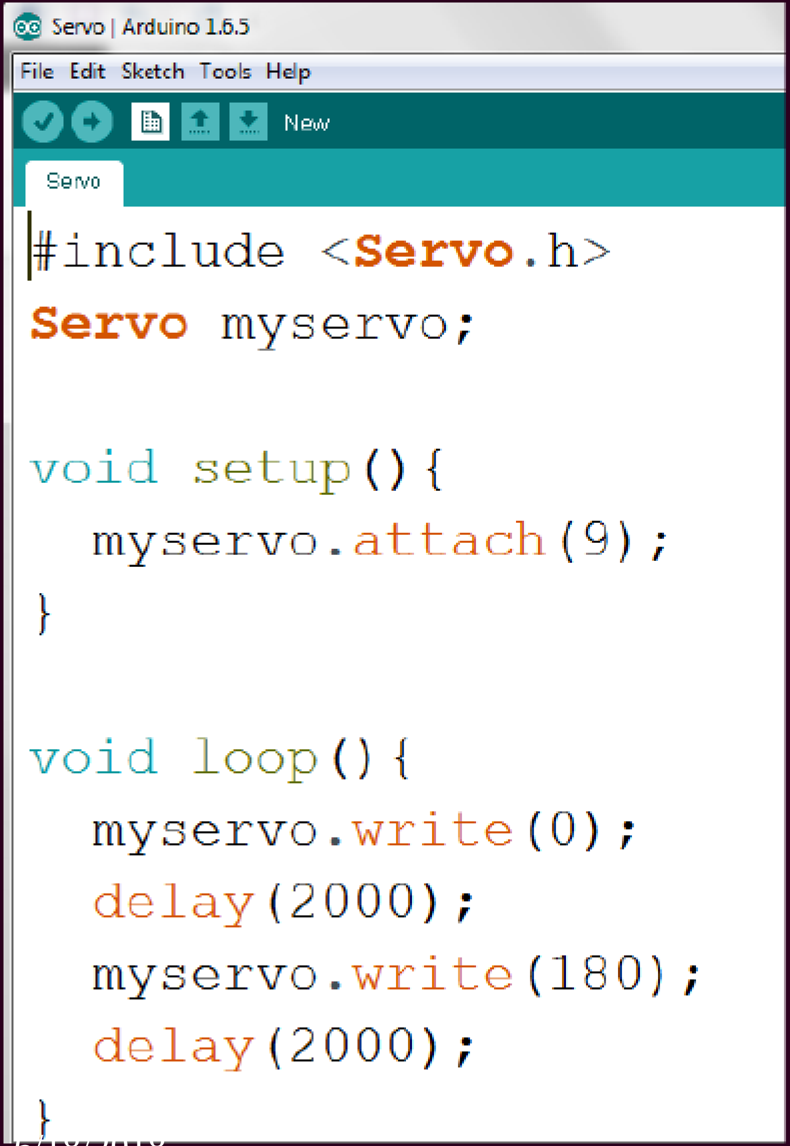
```
my_servo_object.attach(servo_pin);
```

  - Send control signal

```
my_servo_object.write(pos);
```

`attach` and `write` are pre-defined methods that act on the servo object.

# Hands-on Activity: Servo



```
#include <Servo.h>
Servo myservo;

void setup(){
  myservo.attach(9);
}

void loop(){
  myservo.write(0);
  delay(2000);
  myservo.write(180);
  delay(2000);
}
```

- Control two servos

# DC Motor

- Control the speed of DC motor by controlling the input voltage to the motor – i.e. use PWM signal
- PWM allows us to adjust the average value of the input voltage – by turning on and off the power at a specific rate
- Average voltage depends on
  - the duty cycle
  - Amount of time the signal is on vs off in a signal period of time
- Controlling the rotation direction – inverse the direction of the current flow through motors
- Motor driver – improved L298N module

# DC motor cont. L298N Driver

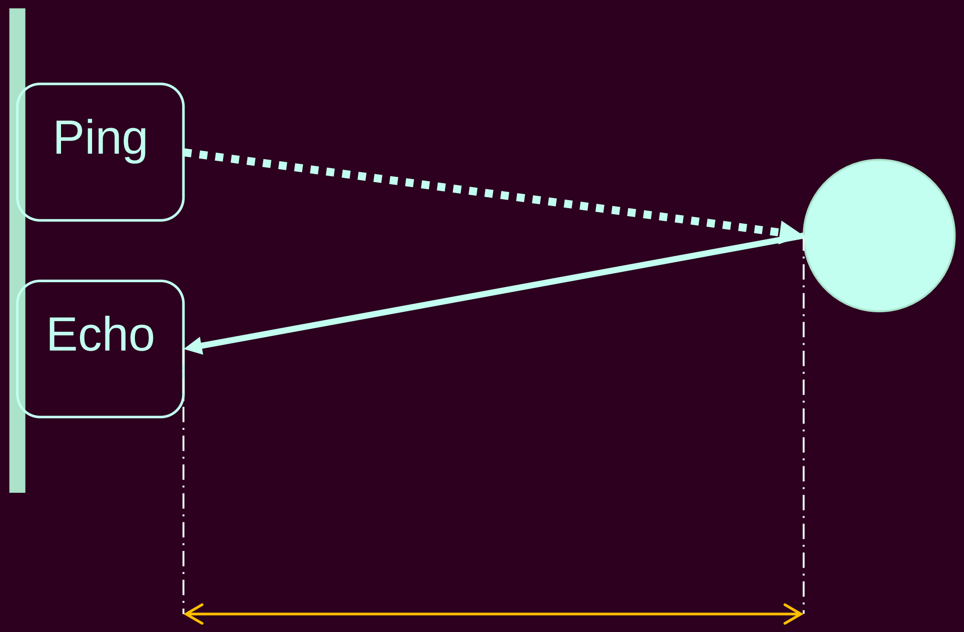
- A dual H-bridge Motor driver
- Allows speed and direction control of two DC motors at the same time
- Using Arduino pins to control

# Ultrasonic

- Emit an ultrasound at 40 KHz
- Bounced back after hitting an obstacle
- Distance = speed x time = speed of sound x travelled time

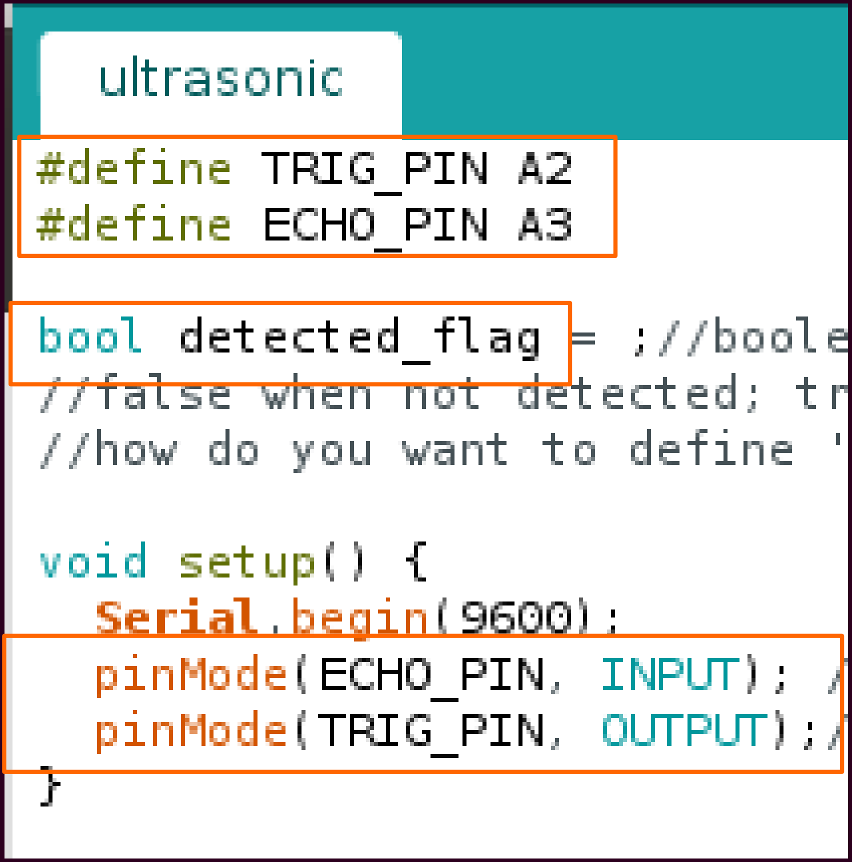
# Ultrasonic

- Source code will return distance



Distance X cm
time taken = total travelled time?

# Ultrasonic

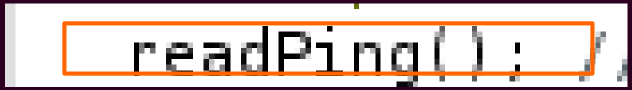


```
#define TRIG_PIN A2
#define ECHO_PIN A3

bool detected_flag = ;//boole
//false when not detected; tr
//how do you want to define '

void setup() {
  Serial.begin(9600);
  pinMode(ECHO_PIN, INPUT); /
  pinMode(TRIG_PIN, OUTPUT);/
}
```

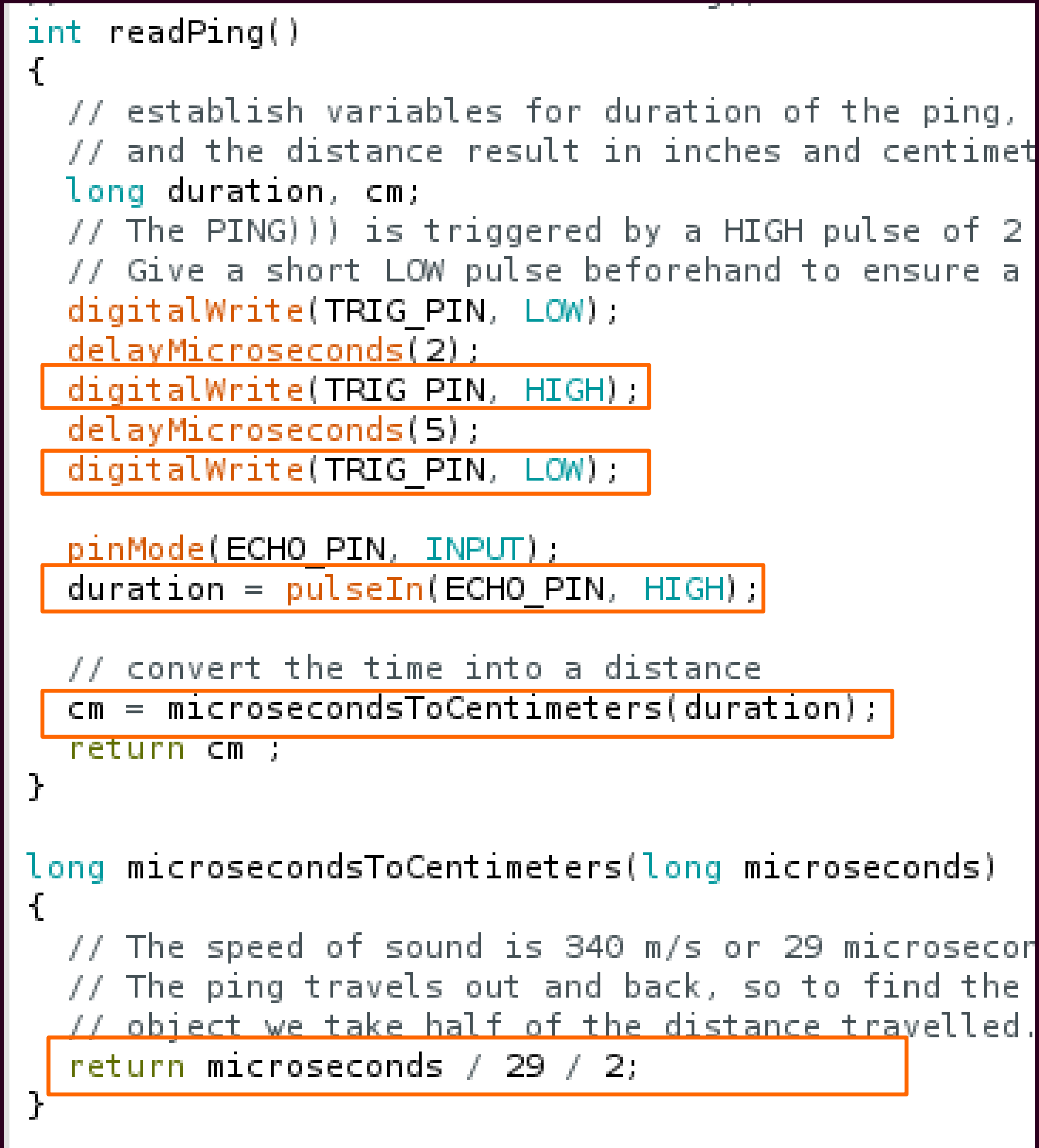
1. Define Trig and Echo pins
2. Pin mode configuration for ECHO and TRIG pins – input or output?



```
  readPing(); /
```

3. readPing() returns distance calculated (cm)

# Ultrasonic



```
int readPing()
{
  // establish variables for duration of the ping,
  // and the distance result in inches and centimet
  long duration, cm;
  // The PING))) is triggered by a HIGH pulse of 2
  // Give a short LOW pulse beforehand to ensure a
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG PIN, HIGH);
  delayMicroseconds(5);
  digitalWrite(TRIG_PIN, LOW);

  pinMode(ECHO_PIN, INPUT);
  duration = pulseIn(ECHO_PIN, HIGH);

  // convert the time into a distance
  cm = microsecondsToCentimeters(duration);
  return cm ;
}

long microsecondsToCentimeters(long microseconds)
{
  // The speed of sound is 340 m/s or 29 microsecon
  // The ping travels out and back, so to find the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}
```

**readPing() returns distance calculated (cm)**

1. Trig sets as HIGH to generate pulse for 10us

2. pulseIn() to read travel time -> duration

3. pulseIn() waits for echo pin to go HIGH and LOW (time travelled – return length of pulse in us)

# Hands-on Activity: Range Sensor

- **ultrasonic.ino** (/sensor folder)
- Try:
  - Display range reading on serial monitor
  - Using 'Flag' concept, define threshold range to raise flag.
  - If flag is raised,  turn on the light. If flag is down, turn off the light.
  - When an object is nearer, blink the light at a higher frequency!
  - Complete, compile and flash the firmware into Arduino

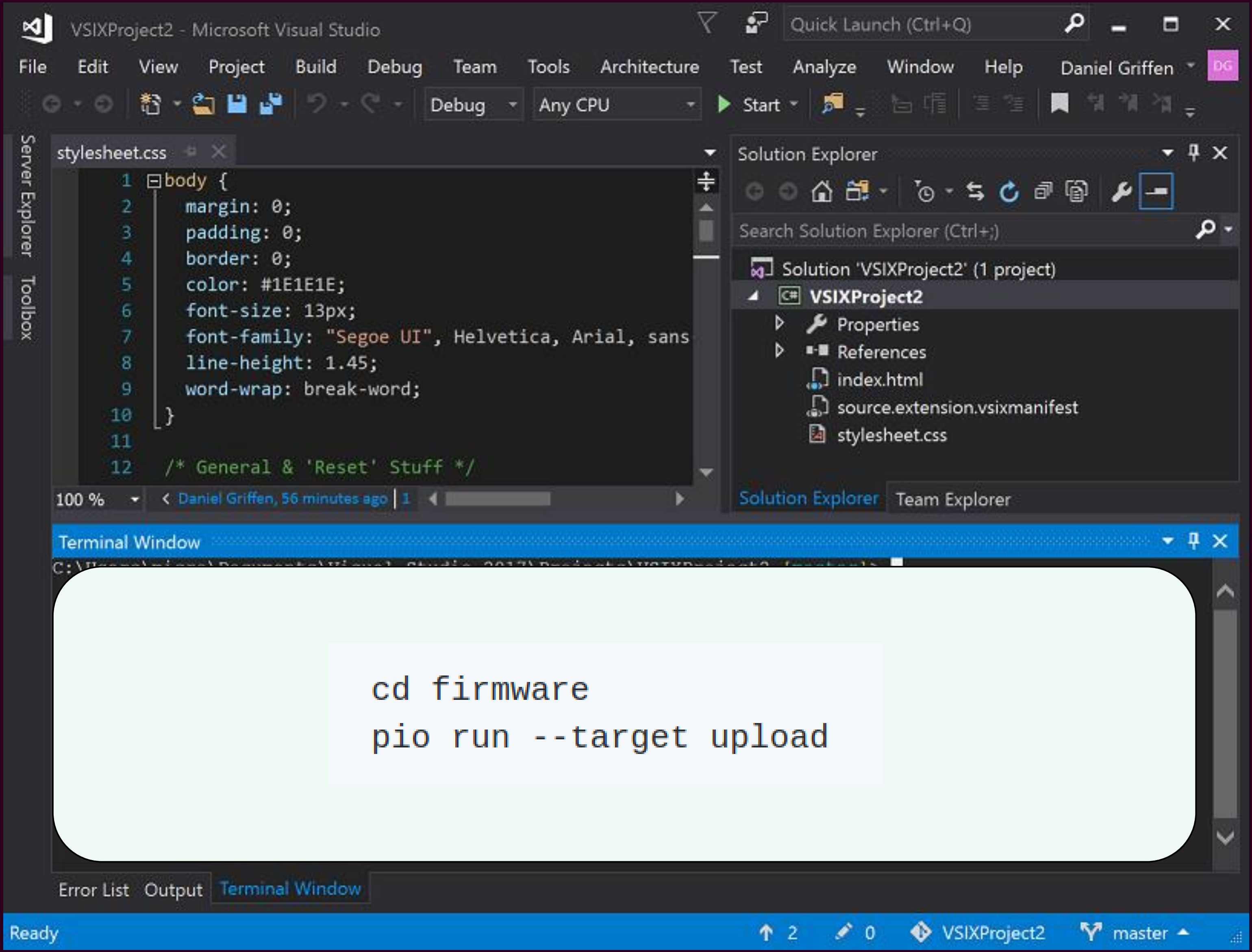
# Hands-on Activity: Servo cont.

- **Servo_SerialComm.ino** (/sensor folder)
- Try:
  - You may recycle your previous servo code!
  - Make servo sweep
  - Control servo with serial command
  - ‘move/sweep’

# Hands-on Activity: DC motor

- **DCMotor.ino** (/sensor folder)
- Try:
  - You may recycle your previous serial code!
  - Set up a server to remote control the DC motors
    - Write a firmware to interface with the server
  - Firmware template
    - https://github.com/grassjelly/robotcar

# Hands-on Activity: Flash in



```css
body {
    margin: 0;
    padding: 0;
    border: 0;
    color: #1E1E1E;
    font-size: 13px;
    font-family: "Segoe UI", Helvetica, Arial, sans
    line-height: 1.45;
    word-wrap: break-word;
}

/* General & 'Reset' Stuff */
```

```
cd firmware
pio run --target upload
```

# server

- Navigate to /sensor/ui folder
- Try:
  - Flash UCrobotics.ino (/sensor folder) again
  - node server.js ( /sensor/ui folder)
  - Control the car from your server!

# reference

- https://www.slideshare.net/jstleger/arduino-101-schuyler-st-leger-desert-code-camp-2012-nov-17
- Arduino cc
- Arduino project hub
- Adafruit