™

ARDUINO

This Evening

- Arduino overview
- Arduino vs Raspberry Pi
- “Hello World” programs
- Input/Output devices
- Speedometer Overview
- Grade Crossing Project

What is Arduino?

The word is an Italian masculine first name meaning "strong friend".

A bar in Ivrea Italy, located in the Northwestern part of the country near Turin.

An Open-Source electronic hardware specification and an Open-Source Software implementation.

5 students attending Interaction Design Institute Ivrea in 2003 lamenting over the price and complexity of obtaining parts to build a robot for a school project. They regularly met in a bar named Arduino.

Scenarios in Model Railroading

Grade Crossing

Dead-End Track Stop

Block Occupancy

Track Crossing Protection

Automation / Animation

Turnout Control

Speedometer

Robotics to Unload a Car

Turn Table Control

Routes

Open and close engine shop doors

Play sounds / recordings / music for specific events

Measure tractive effort of locomotives. “*Strain Gauge*”

Monitor movement of rolling stock *“RFID”*

Emulate a DCC Decoder - *Iowa Scaled Engineering Shield $30*

Lighting *“Check out YouTube”*

- Control an entire town
- Lights within a building
- Streetlights / Yard lights
- Realistic campfire

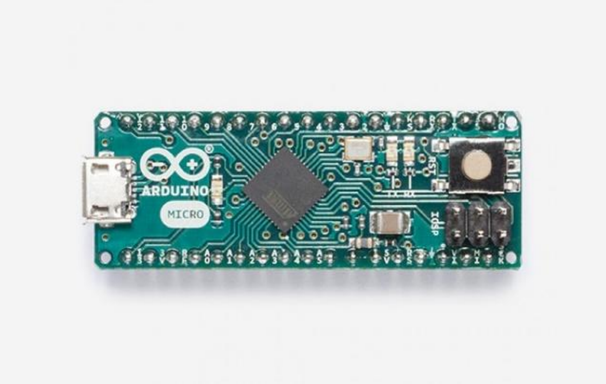
Arduino Boards

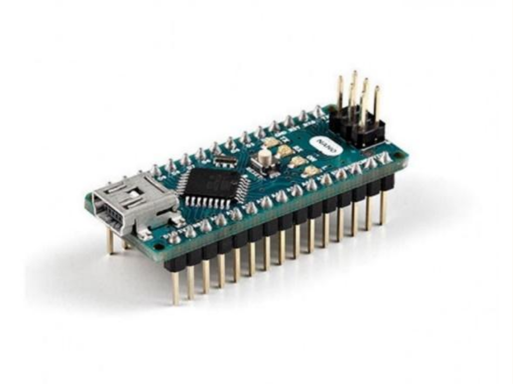
Processors

Board Input Power

Shields

4 Examples of 22 Boards





Arduino Board

Processors

Board Input Power

Shields

Most boards use AVR or ARM processors.

- Some compatible boards are RISC-5.

Migration toward IoT with more boards having on-board communication.

“The Arduino UNO R4 Boards utilize the Renesas RA4M1 processor, which is a 32-bit ARM Cortex-M4 with a dedicated SIMD (Single Instruction,Multiple Data) engine for handling vector operations. This allows for efficient processing of data arrays and other vector-based calculations.”

Arduino Board

Processsors

Board Input Power

Shields

9-volt battery

or

USB Cable

or

A/C Adapter

Boards have different power requirements.

7 -> 12 volt is common, some require only 5 volts.

Some boards operate at 3.3 volts, others at 5 volts.

VIN input pin with internal regulator. > 6v

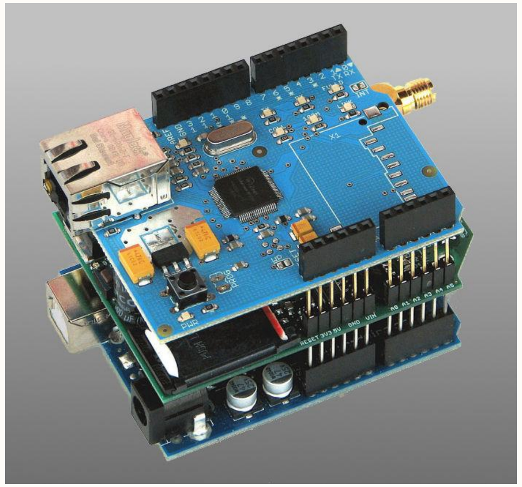
Regulated input/output pin.

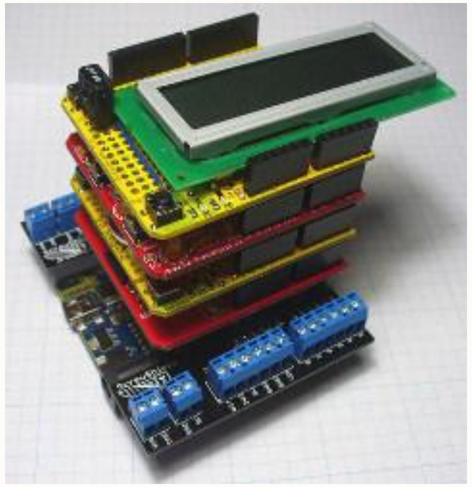
Arduino Board

Processors

Board Input Power

Shields





Shields add specific functionality.

i.e. motor, WiFi, robotics

Most Shields are pluggable.

Hundreds of shields are available today.

Arduino Board

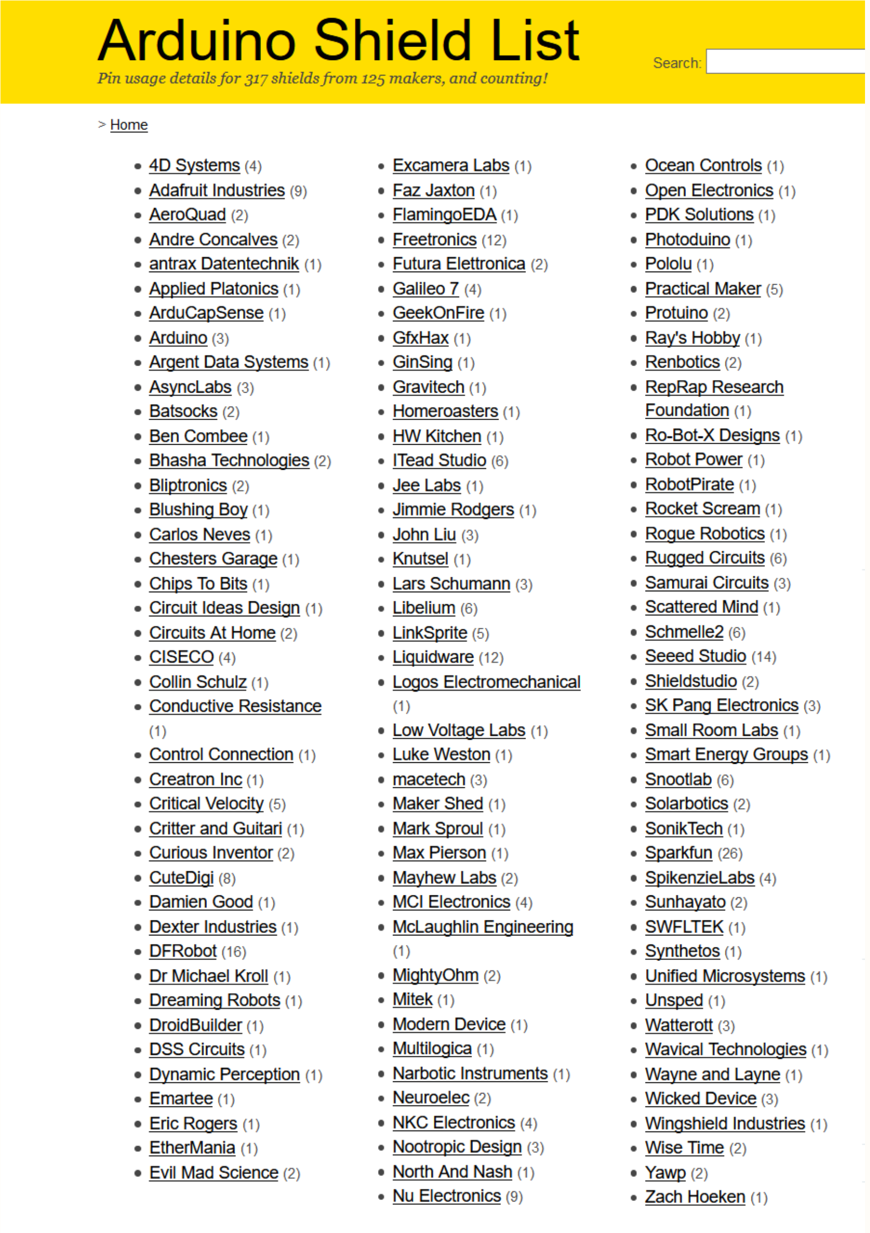
Processors

Board Input Power

Shields

“317” Shields –
From 125 manufactures
as of 2025\05

http://shieldlist.org/



Arduino Shield List

Pin usage details for 317 shields from 125 makers, and counting!

Search:

> Home

- 4D Systems (4)
- Adafruit Industries (9)
- AeroQuad (2)
- Andre Concalves (2)
- antrax Datentechnik (1)
- Applied Platonics (1)
- ArduCapSense (1)
- Arduino (3)
- Argent Data Systems (1)
- AsyncLabs (3)
- Batsocks (2)
- Ben Combee (1)
- Bhasha Technologies (2)
- Bliptronics (2)
- Blushing Boy (1)
- Carlos Neves (1)
- Chesters Garage (1)
- Chips To Bits (1)
- Circuit Ideas Design (1)
- Circuits At Home (2)
- CISECO (4)
- Collin Schulz (1)
- Conductive Resistance (1)
- Control Connection (1)
- Creatron Inc (1)
- Critical Velocity (5)
- Critter and Guitari (1)
- Curious Inventor (2)
- CuteDigi (8)
- Damien Good (1)
- Dexter Industries (1)
- DFRobot (16)
- Dr Michael Kroll (1)
- Dreaming Robots (1)
- DroidBuilder (1)
- DSS Circuits (1)
- Dynamic Perception (1)
- Emartee (1)
- Eric Rogers (1)
- EtherMania (1)
- Evil Mad Science (2)
- Excamera Labs (1)
- Faz Jaxton (1)
- FlamingoEDA (1)
- Freetronics (12)
- Futura Elettronica (2)
- Galileo 7 (4)
- GeekOnFire (1)
- GfxHax (1)
- GinSing (1)
- Gravitech (1)
- Homeroasters (1)
- HW Kitchen (1)
- ITead Studio (6)
- Jee Labs (1)
- Jimmie Rodgers (1)
- John Liu (3)
- Knutsel (1)
- Lars Schumann (3)
- Libelium (6)
- LinkSprite (5)
- Liquidware (12)
- Logos Electromechanical (1)
- Low Voltage Labs (1)
- Luke Weston (1)
- macetech (3)
- Maker Shed (1)
- Mark Sproul (1)
- Max Pierson (1)
- Mayhew Labs (2)
- MCI Electronics (4)
- McLaughlin Engineering (1)
- MightyOhm (2)
- Mitek (1)
- Modern Device (1)
- Multilogica (1)
- Narbotic Instruments (1)
- Neuroelec (2)
- NKC Electronics (4)
- Nootropic Design (3)
- North And Nash (1)
- Nu Electronics (9)
- Ocean Controls (1)
- Open Electronics (1)
- PDK Solutions (1)
- Photoduino (1)
- Pololu (1)
- Practical Maker (5)
- Protuino (2)
- Ray's Hobby (1)
- Renbotics (2)
- RepRap Research Foundation (1)
- Ro-Bot-X Designs (1)
- Robot Power (1)
- RobotPirate (1)
- Rocket Scream (1)
- Rogue Robotics (1)
- Rugged Circuits (6)
- Samurai Circuits (3)
- Scattered Mind (1)
- Schmelle2 (6)
- Seeed Studio (14)
- Shieldstudio (2)
- SK Pang Electronics (3)
- Small Room Labs (1)
- Smart Energy Groups (1)
- Snootlab (6)
- Solarbotics (2)
- SonikTech (1)
- Sparkfun (26)
- SpikenzieLabs (4)
- Sunhayato (2)
- SWFLTEK (1)
- Synthetos (1)
- Unified Microsystems (1)
- Unsped (1)
- Watterott (3)
- Wavical Technologies (1)
- Wayne and Layne (1)
- Wicked Device (3)
- Wingshield Industries (1)
- Wise Time (2)
- Yawp (2)
- Zach Hoeken (1)

Arduino

The Arduino is a microcontroller. It is not a general-purpose computer.

- It is designed to run software to do a specific task, such as controlling a garage door opener or a microwave oven.
- In fact, it is very similar to a DCC decoder. It can be used to control turnouts, signals, read RFID tags, and a number of layout automation and DCC/LCC tasks.
- As with the RPi, it also has a number of "shields" that add functions and input/output capabilities such as communication, and motor control boards, etc.
- It has a dedicated IDE which you can download for free to aid in programming.

Raspberry Pi

The Raspberry Pi is a complete computer.

- Requires an Operating System.
- More powerful than the legendary Commodore 64, but at a lot less cost.
- It can function as a stand-alone computer, or you can use it to run JMRI, as well as other tasks.
- It was developed for the purpose of teaching programming and supports numerous languages.
- A number of accessories “pHats” are available for specific tasks and to expand the computer beyond its basic configuration.

Sketch

Sketch is the name given to an Ardunio program.

Uses the C/C++ programming language. With limitations.

- Most limitations related to the sketch being written for a micro controller.

Many libraries are available.

A free IDE (Integrated Development Environment) is available to edit, compile, link and load a sketch.

Four basic parts of every sketch are:

- Includes
- Constants
- setup()
- loop()

There is no multi-process, nor multi-threading, support in the ATmega processors.

Simple

“Hello World”

example:

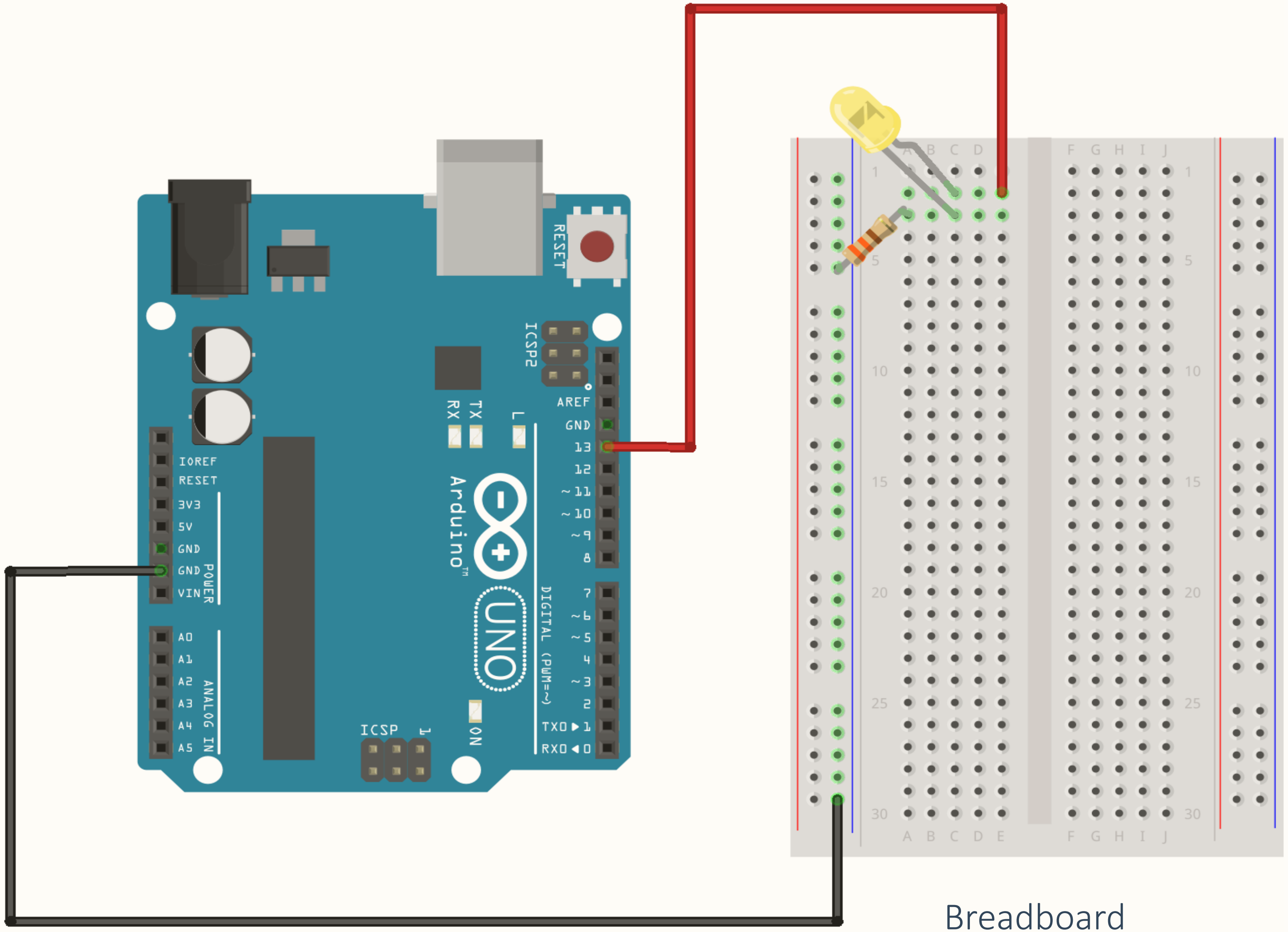
Attach an LED

and

Have it blink at 1 second intervals

Wiring Schematic

For Blink “Hello World”



Breadboard

Sample Sketch

For Blink “Hello World”

```
/*  Blink -  Turns on an LED on for one second, then off for one second,
repeatedly.   This example code is in the public domain. */

int led1 = 13;     // Give it a name and indicate what pin number to use

//  The setup routine runs once at startup or when you press reset:
void setup() {
 pinMode( led1, OUTPUT );    // initialize the digital pin as an output.
}

// The loop routine runs over and over again forever:
void loop() {
  digitalWrite( led1, HIGH );   // turn the LED on (HIGH is the voltage level)
  delay( 1000 );                // wait for a second
  digitalWrite( led1, LOW );    // turn the LED off by making the voltage LOW
  delay( 1000 );                // wait for a second
}
```

Fading LED Sketch

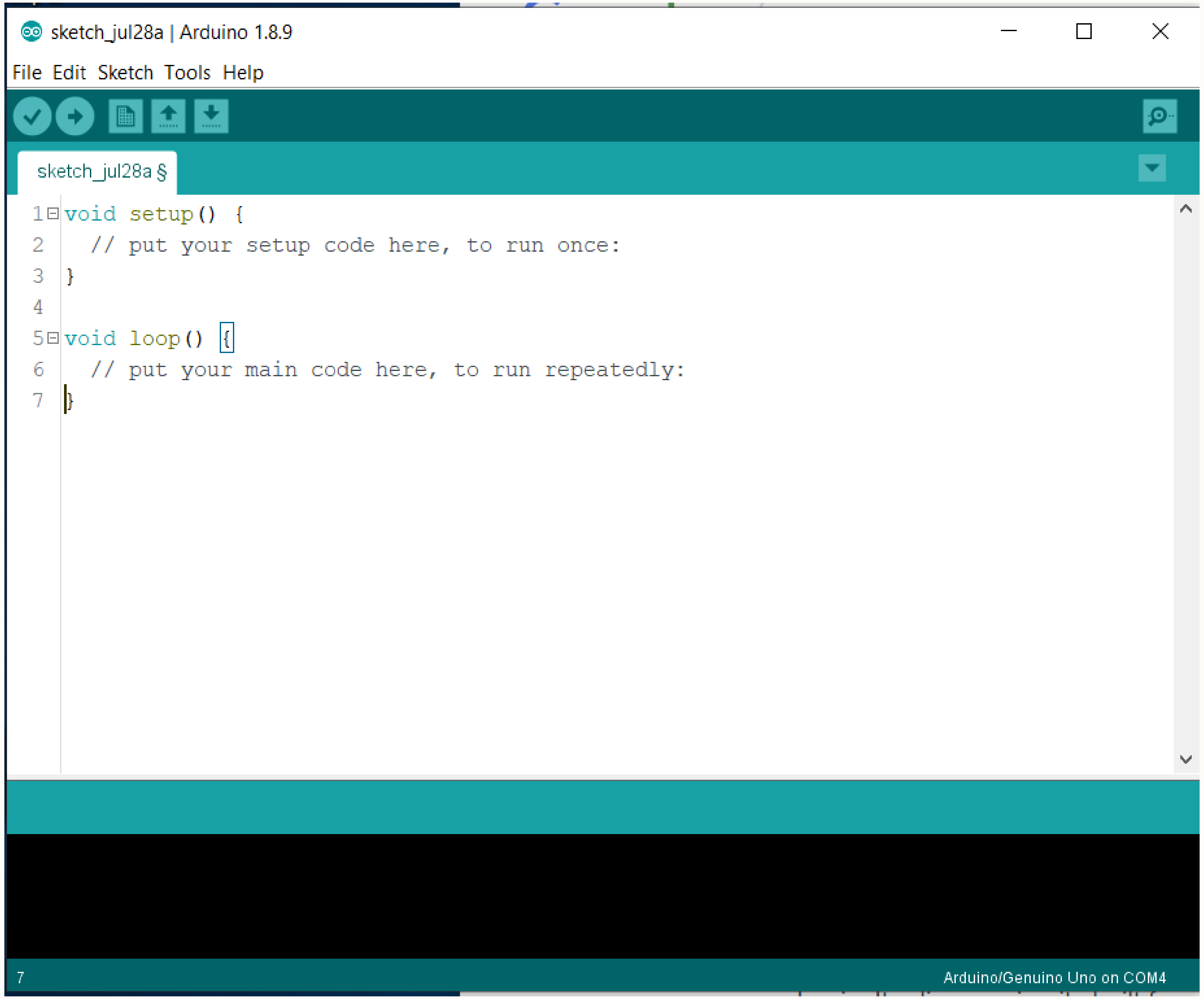
```
int beacon = 9;            //  The PWM pin
int brightness = 125;   // Start at halfway    Range is  Zero -> 255
int increment = 5;        // The value to increment the brightness

void setup() {
 pinMode( beacon, OUTPUT );    // initialize the PWM pin as an output.
}

void loop() {
  brightness = brightness + increment;

  if ( brightness > 255 ) {
    brightness = 255;
    increment = -5;
   } else if (brightness < 0 ) {
    brightness = 0;
    increment = 5;
   }

  analogWrite( beacon,  brightness  );
  delay( 250 );   // wait a quarter second
}
```

Fading LED Sketch

```
…
void setup() {
 Serial.begin(9600);
 pinMode( beacon, OUTPUT );    // initialize the PWM pin as an output.
}

void loop() {
  brightness = brightness + increment;

  if ( brightness > 255 ) {
    brightness = 255;
    increment = -5;
   } else if (brightness < 0 ) {
    brightness = 0;
    increment = 5;
   }

  Serial.print(“Brightness value is  ”);
  Serial.println(brightness);

   analogWrite( beacon,  brightness  );
  delay( 250 );   // wait a quarter second
}
```

Simple, free IDE

Integrated Development Environment

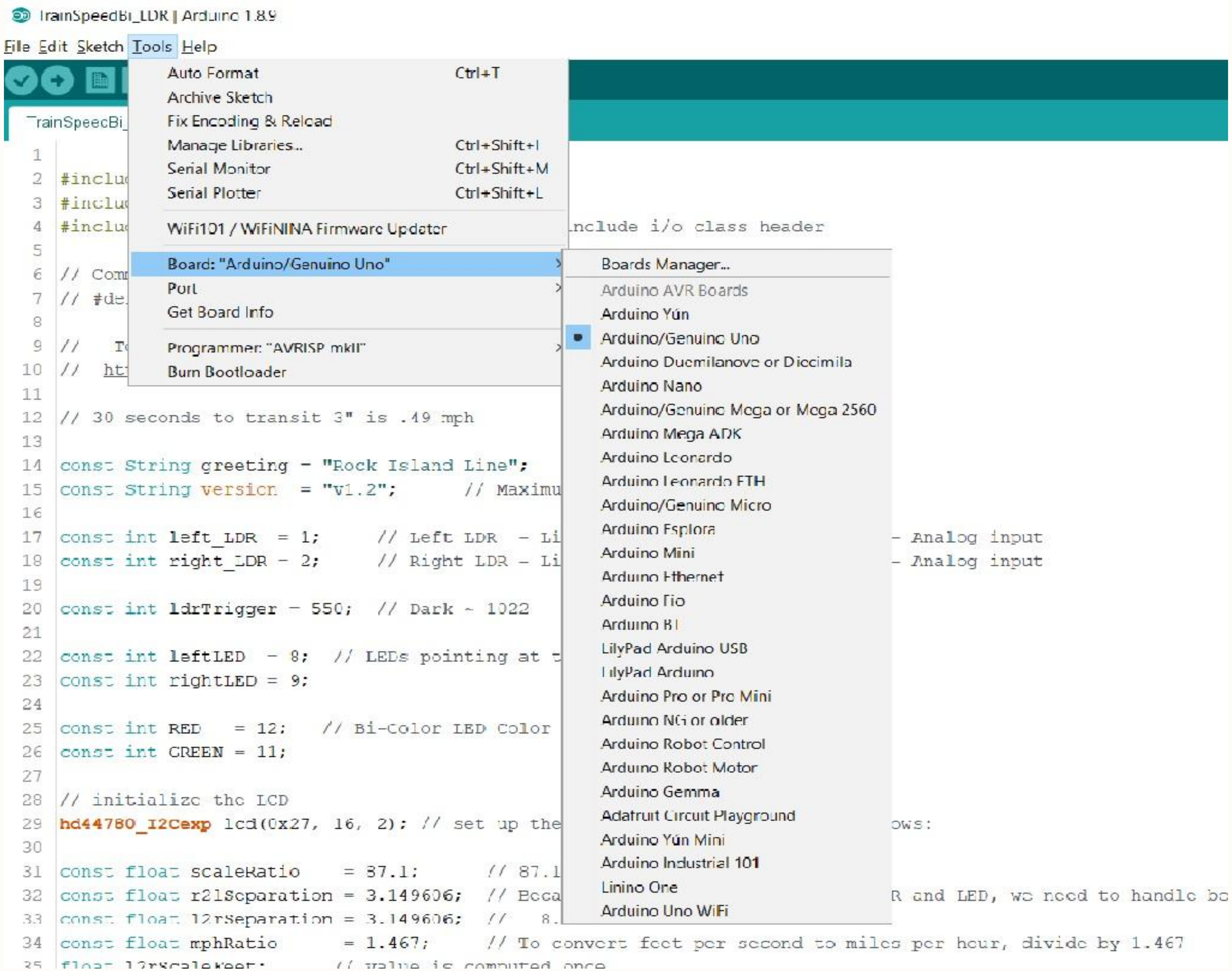

sketch_jul28a | Arduino 1.8.9
File Edit Sketch Tools Help
sketch_jul28a §
7
Arduino/Genuino Uno on COM4

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Specify the board in the IDE

Needs to know how to communicate with each board

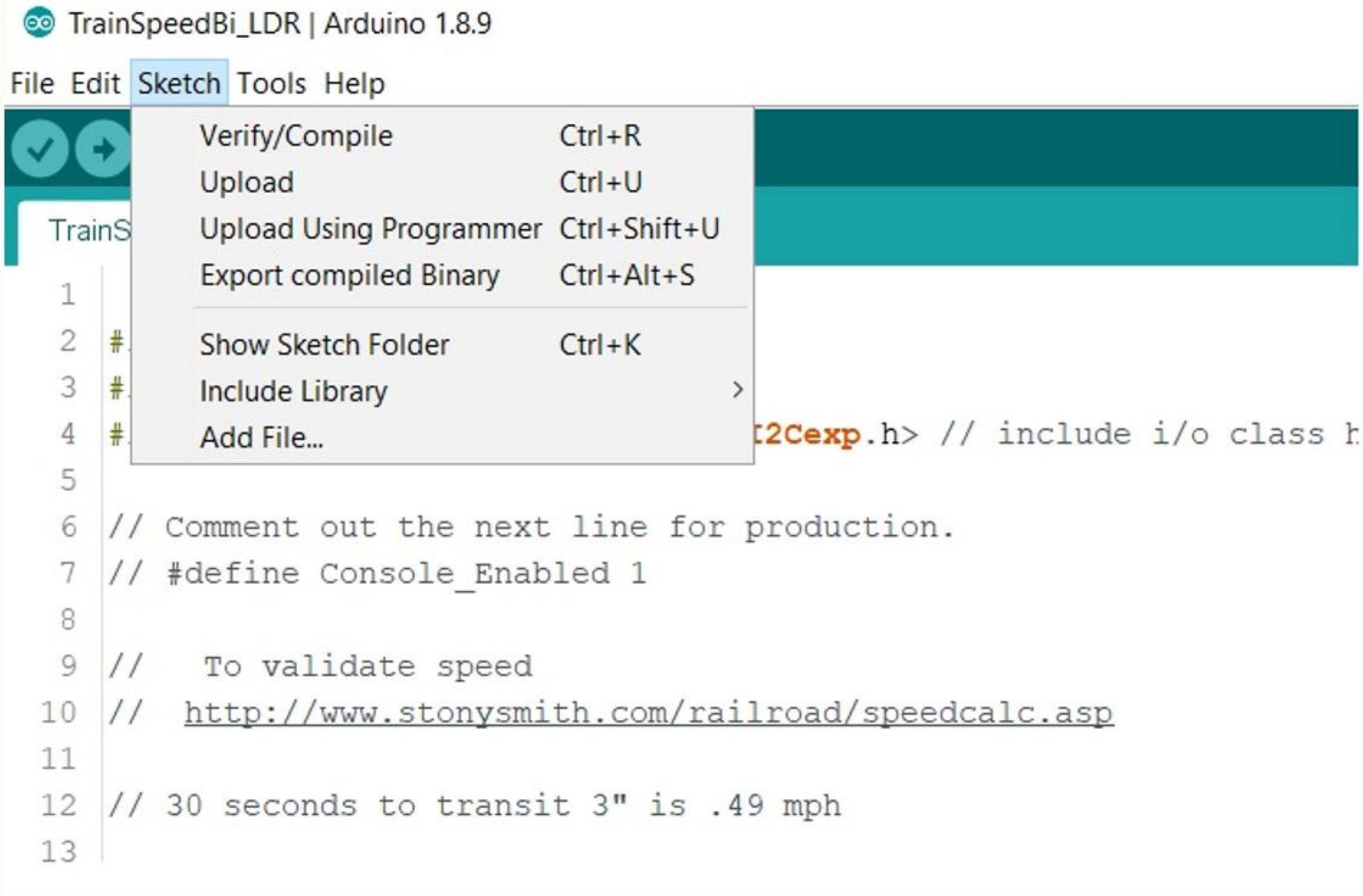
Needs to know which board is connected



```

#inclu
#inclu
#inclu                                                     nclude i/o class header

// Comm
// #de

//    To
//  ht

// 30 seconds to transit 3" is .49 mph

const String greeting = "Rock Island Line";
const String version  = "v1.2";       // Maximu

const int left_LDR  = 1;      // Left LDR  - Li                          - Analog input
const int right_LDR = 2;      // Right LDR - Li                          - Analog input

const int ldrTrigger = 550;  // Dark ~ 1022

const int leftLED  = 8;  // LEDs pointing at t
const int rightLED = 9;

const int RED   = 12;   // Bi-Color LED Color
const int GREEN = 11;

// initialize the LCD
hd44780_I2Cexp lcd(0x27, 16, 2); // set up the                            ows:

const float scaleRatio    = 87.1;      // 87.1
const float r2lSeparation = 3.149606;  // Beca                            R and LED, we need to handle bo
const float l2rSeparation = 3.149606;  //   8.
const float mphRatio      = 1.467;     // To convert feet per second to miles per hour, divide by 1.467
float l2rScaleFeet;        // value is computed once
```

IDE Compile and load

One tool to create sketches, and load them onto the board

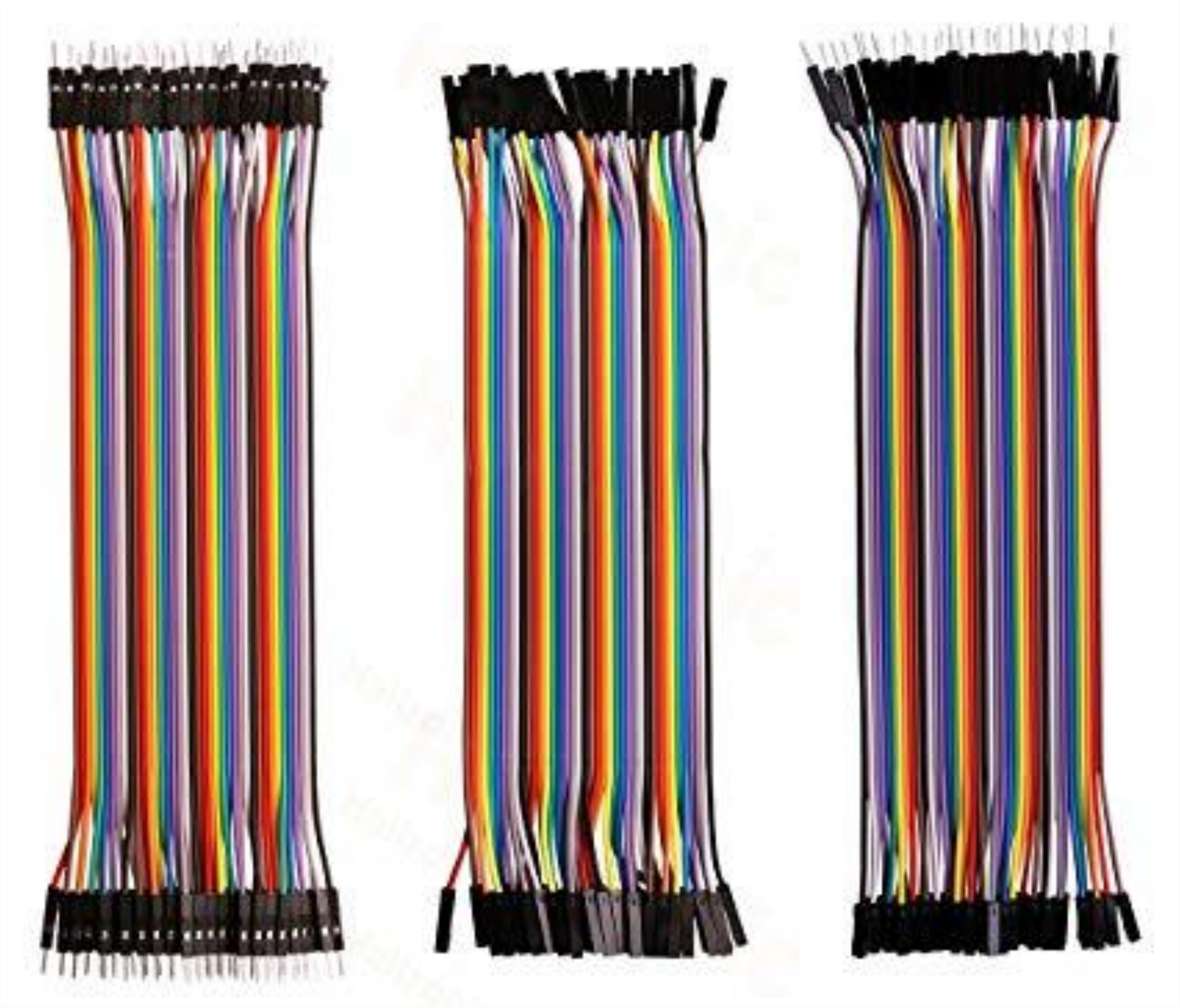

TrainSpeedBi_LDR | Arduino 1.8.9
File Edit Sketch Tools Help
TrainS
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library ›
Add File...

```

#
#
#                         2Cexp.h> // include i/o class h

// Comment out the next line for production.
// #define Console_Enabled 1

//   To validate speed
//  http://www.stonysmith.com/railroad/speedcalc.asp

// 30 seconds to transit 3" is .49 mph

```

Three types of jumper wires

Varying lengths are available

You’re gona need all three



Male to Male

Female to Female

Male to Female

Individual Devices and Capabilities…

A few considerations…

Is the device Input or Output or Both?

Analog or Digital?

PWM - Pulse Width Modulation

Do we need to use Pull-Up or Pull-Down for the input device?

Maximum pin output current is ~ 40 milliamps.

R4 UNO pin limit is 8 milliamps.

Coming Up:

Grouped into Input & Output devices.

Here are a few samples…

Input devices

Sound Sensors

Range Finder

Light Sensor

Motion/IR Sensors

Position Sensors

Weather Sensors

Bio Sensors

RFID

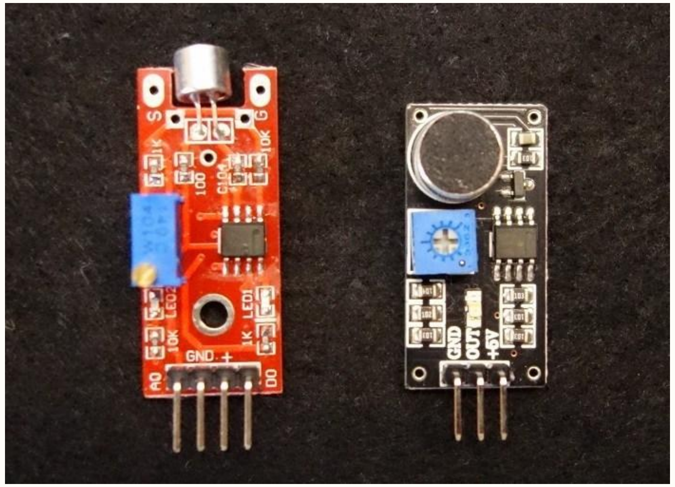
Remote Control

Joy Sticks

KeyPads

Real-Time Clock

Sound Sensors aka Microphone

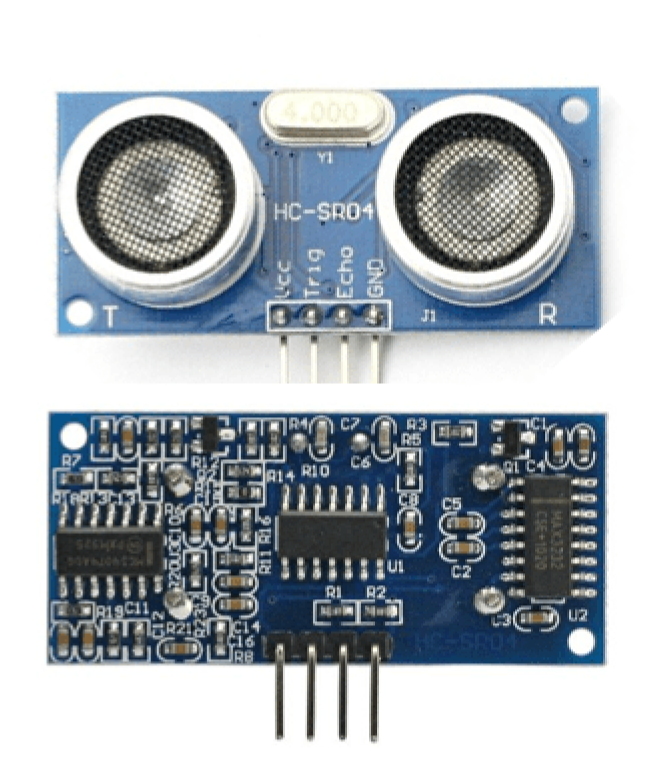


Speech Recognition Shield

Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- KeyPads
- Real-Time Clock

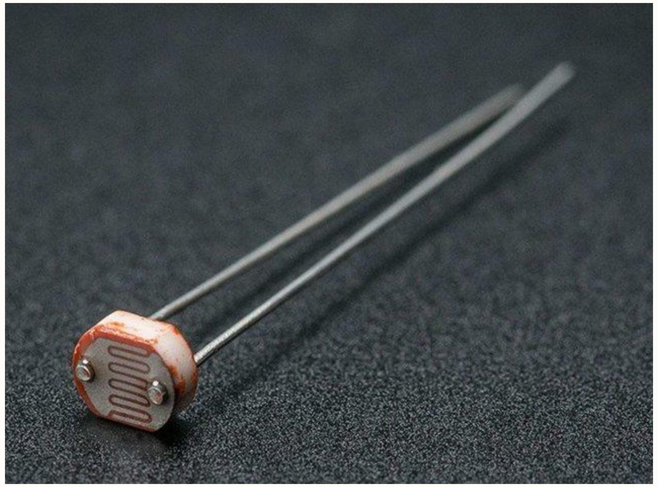
Ultrasonic Sensor Range Finder



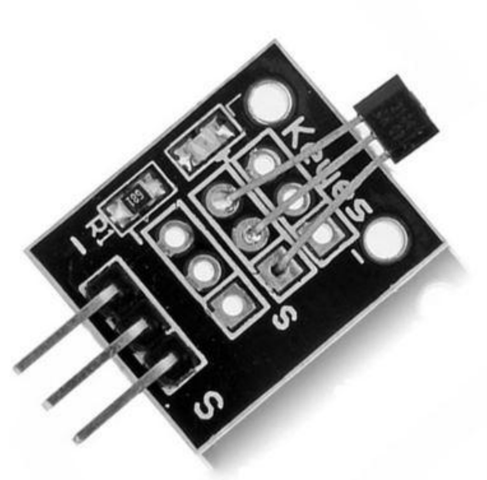
Input devices

- Sound Sensors
- Range Finder
- **Light Sensor**
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- KeyPads
- Real-Time Clock

Photocell - Indicates the amount of light.



Hall Effect Sensor



Input devices

Sound Sensors

Range Finder

Light Sensor

Motion/IR Sensors

Position Sensors

Weather Sensors

Bio Sensors

RFID

Remote Control

Joy Sticks

KeyPads

Real-Time Clock

Motion Sensor

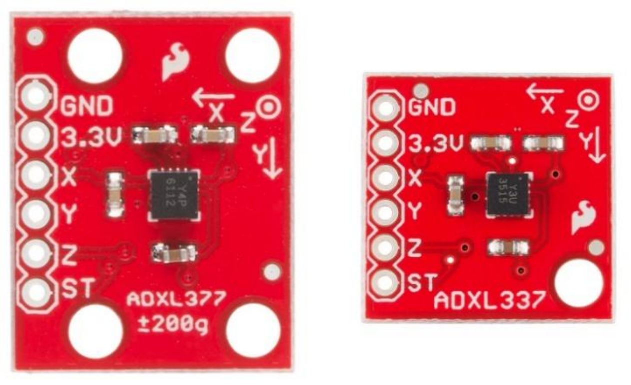
Infrared Detector

IR Collision Avoidance

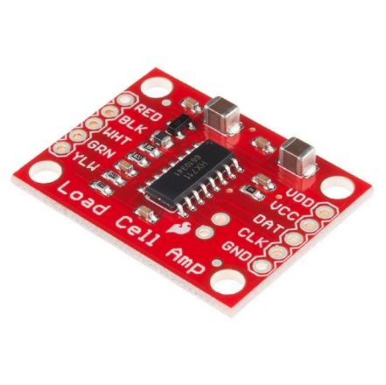
Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- Keypads
- Real-Time Clock

3 Axis Accelerometers

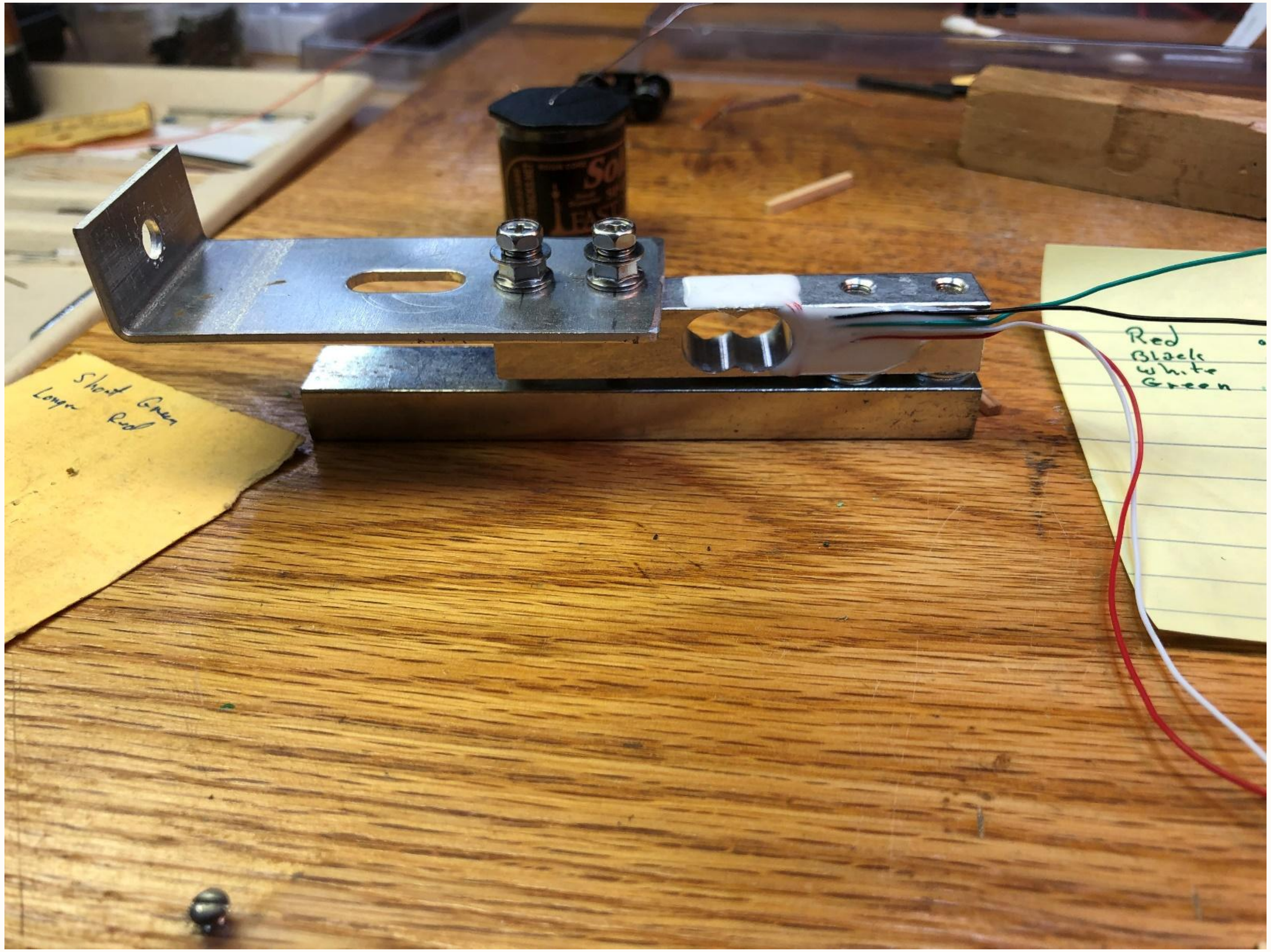


Strain Gauge



Angle Sensor

Business of end of the strain gauge.

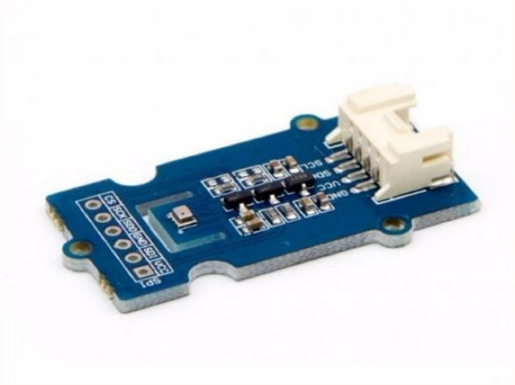


Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- Keypads
- Real-Time Clock

Water Sensor

Barometer / Altimeter / Temperature



Temperature and Humidity Sensor

Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- **Bio Sensors**
- RFID
- Remote Control
- Joy Sticks
- Keypads
- Real-Time Clock

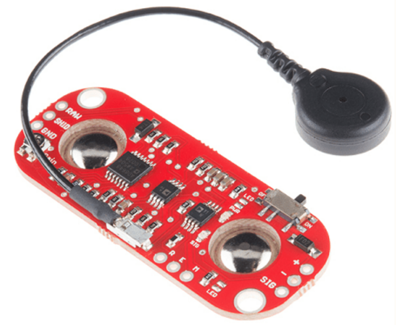
Galvanic Skin Sensor

Finger Print Reader

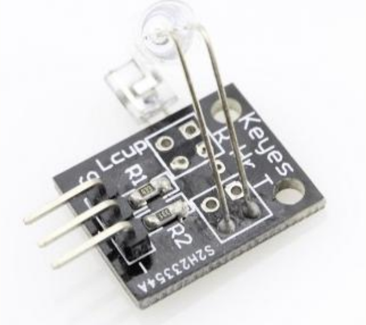
Pulse Sensor

Alcohol Sensor

Muscle Sensor



Finger Heart Rate Sensor

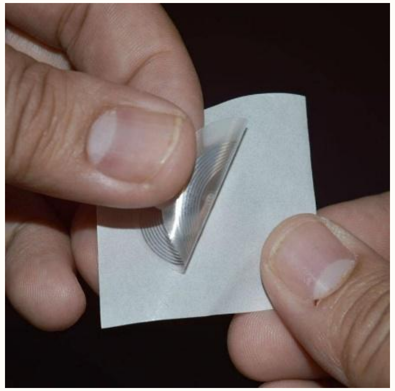


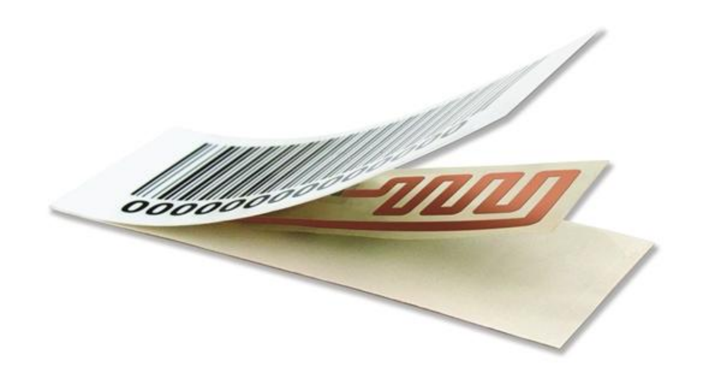
Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- Keypads
- Real-Time Clock

RFID Reader

Passive RFID Tags

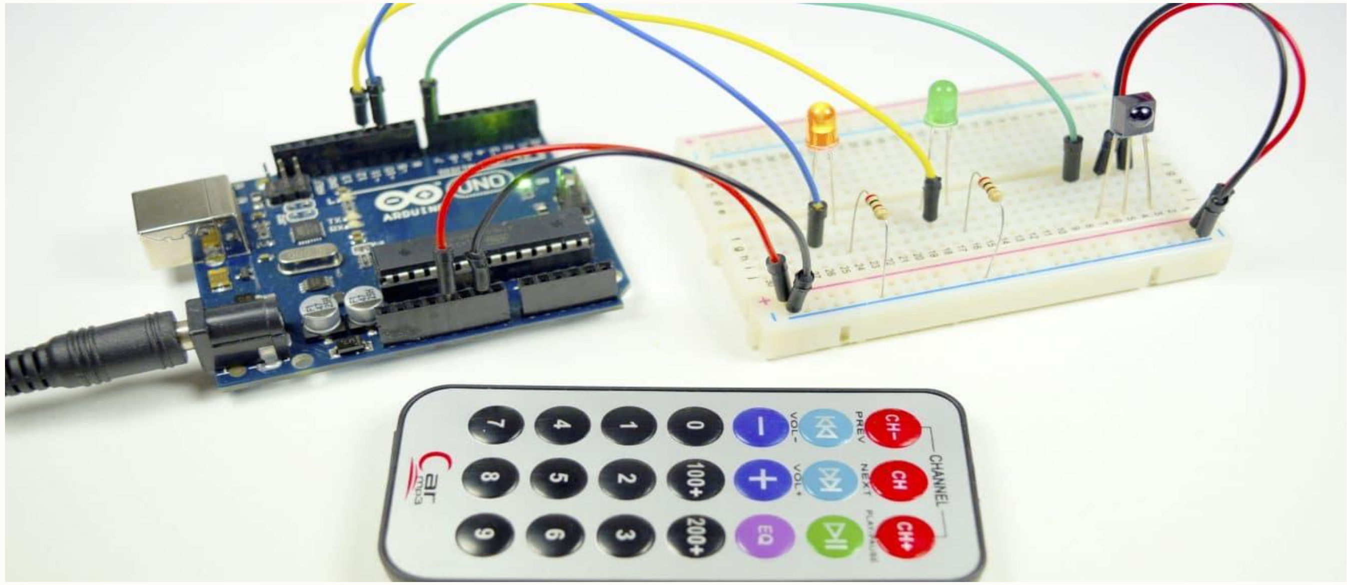




Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- Keypads
- Real-Time Clock

Remote Control



Input devices

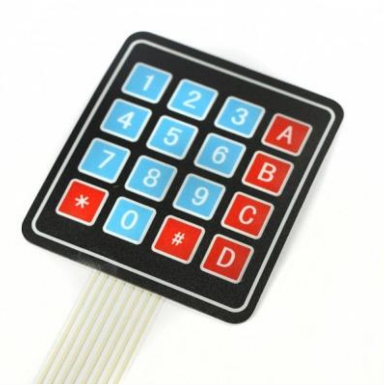
- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- Keypads
- Real-Time Clock

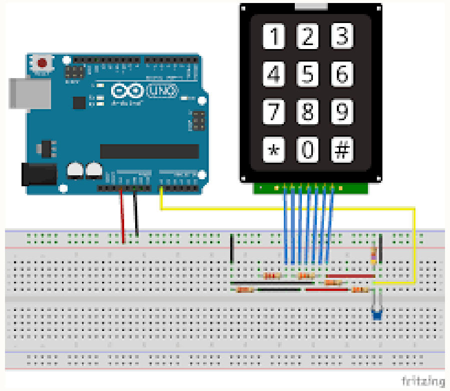
Joy Sticks

Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- **Keypads**
- Real-Time Clock

Key Pads

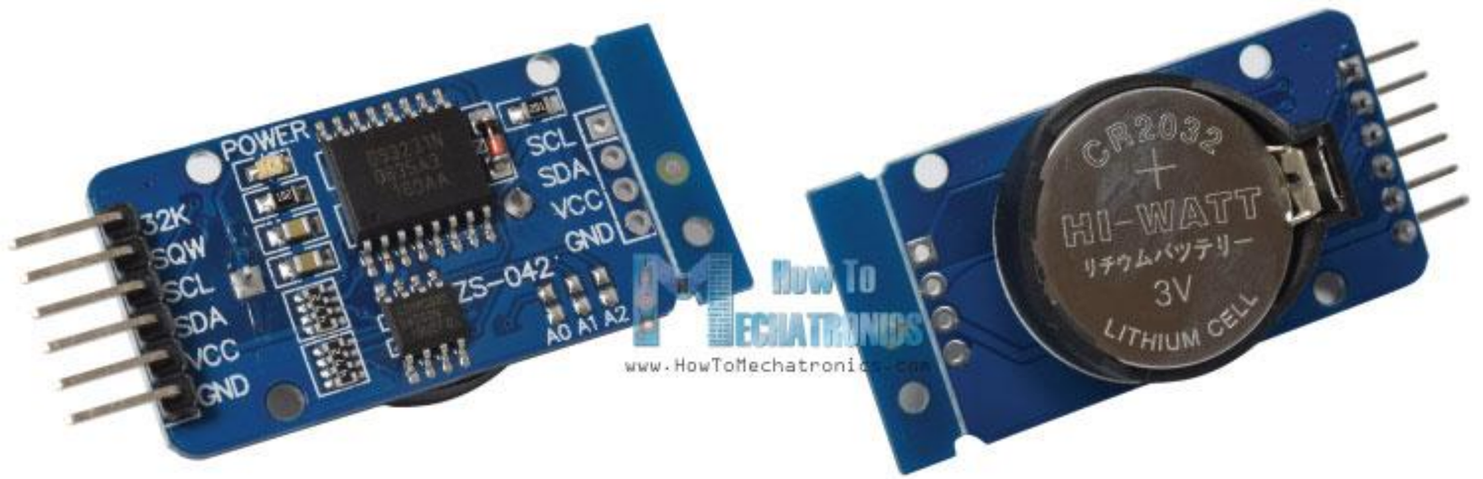




Input devices

- Sound Sensors
- Range Finder
- Light Sensor
- Motion/IR Sensors
- Position Sensors
- Weather Sensors
- Bio Sensors
- RFID
- Remote Control
- Joy Sticks
- Keypads
- **Real-Time Clock**

Real Time Clock



Output Devices

LEDs

Speakers / Buzzers

Communications

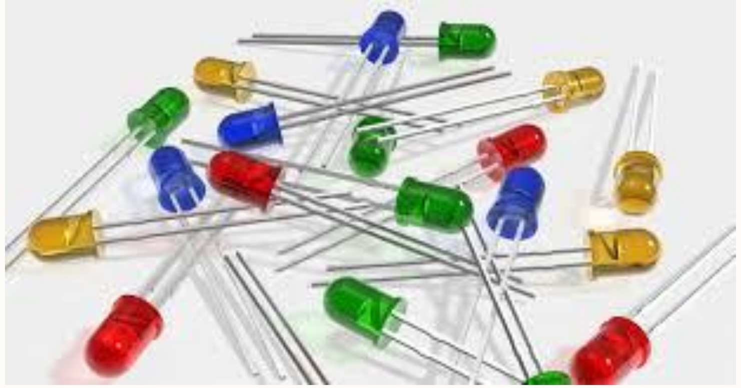
7-Segment Display

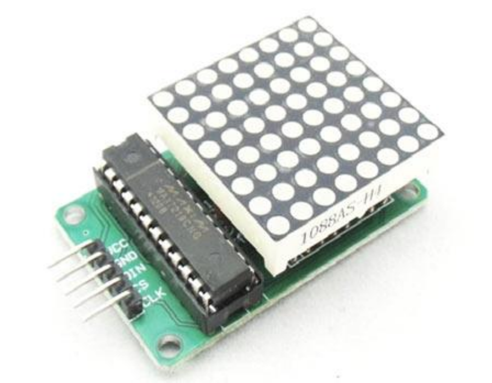
LCD Display

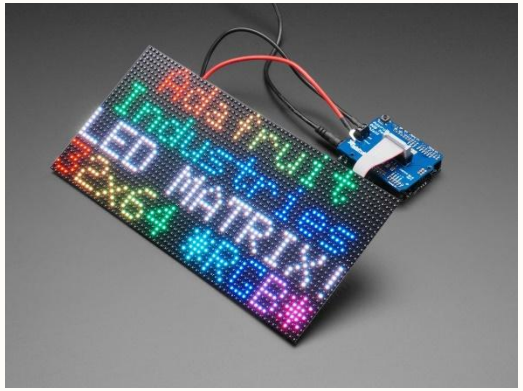
Motors

SPDT-Relay

LEDs



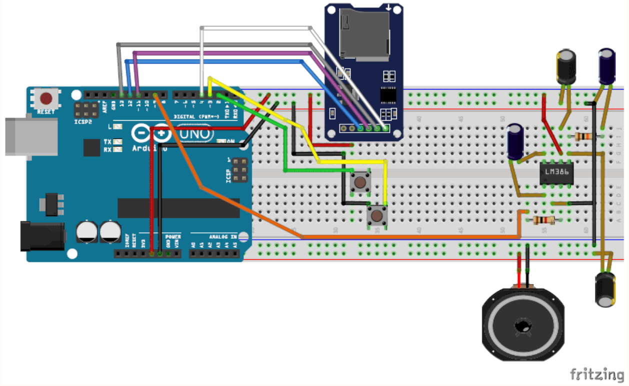




Output Devices

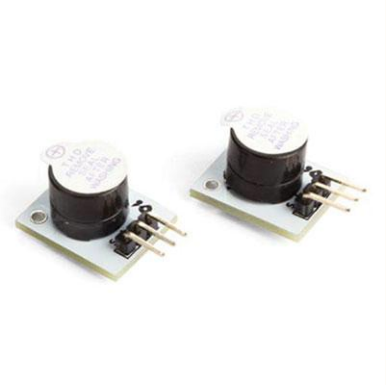
- LEDs
- **Speakers / Buzzers**
- Communications
- 7-Segment Display
- LCD Display
- Motors
- SPDT-Relay

Speakers



Passive Buzzer

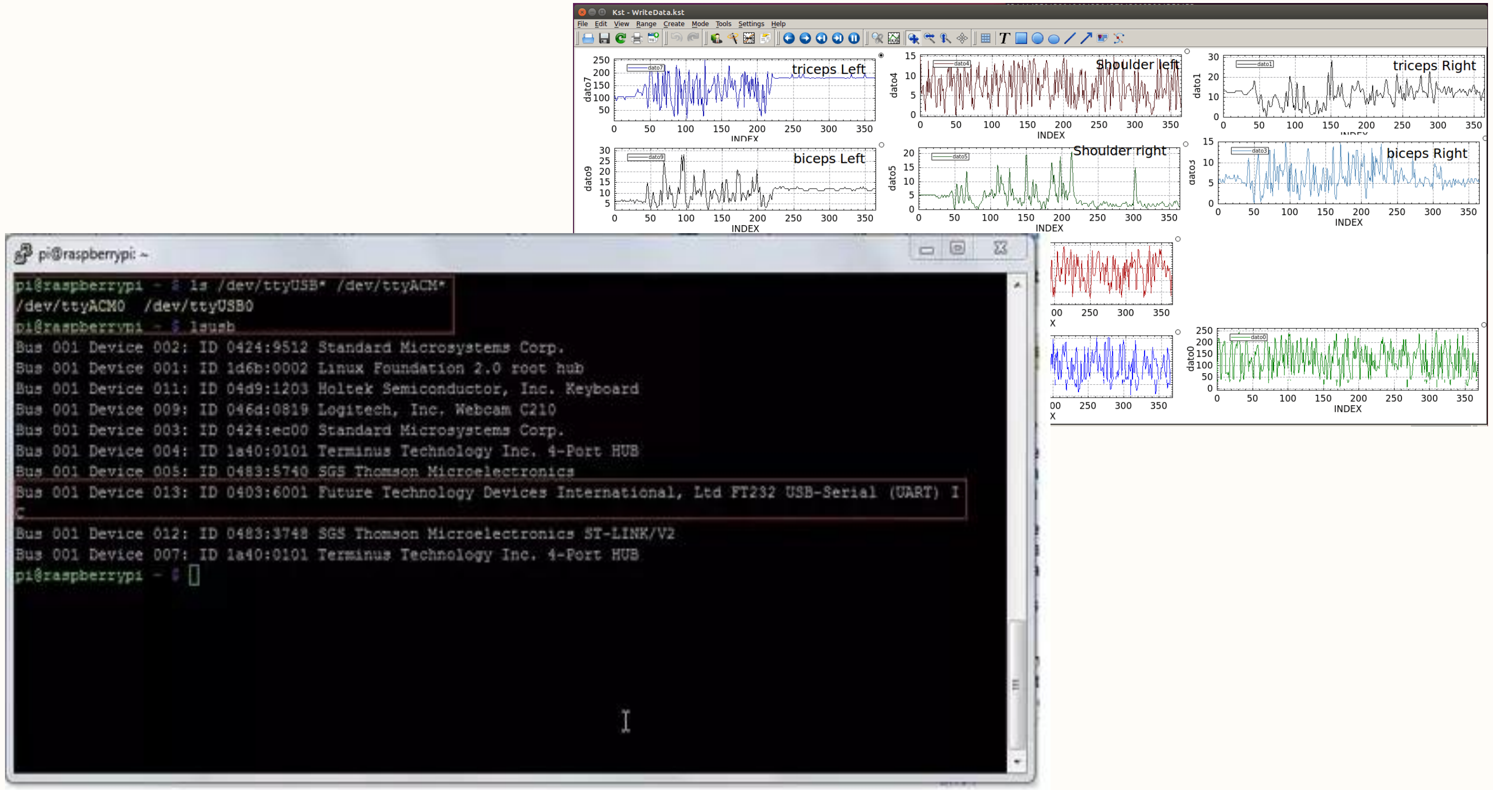
Active Buzzer



Communication

To a Host or another Arduino or mobile device

TCP/IP, Serial Port, WiFi, USB, Bluetooth, Radio, CAN



```
pi@raspberrypi ~ $ ls /dev/ttyUSB* /dev/ttyACM*
/dev/ttyACM0  /dev/ttyUSB0
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 011: ID 04d9:1203 Holtek Semiconductor, Inc. Keyboard
Bus 001 Device 009: ID 046d:0819 Logitech, Inc. Webcam C210
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 1a40:0101 Terminus Technology Inc. 4-Port HUB
Bus 001 Device 005: ID 0483:5740 SGS Thomson Microelectronics
Bus 001 Device 013: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) I
C
Bus 001 Device 012: ID 0483:3748 SGS Thomson Microelectronics ST-LINK/V2
Bus 001 Device 007: ID 1a40:0101 Terminus Technology Inc. 4-Port HUB
pi@raspberrypi ~ $ 
```

Output Devices

- LEDs
- Speakers / Buzzers
- Communications
- 7-Segment Display
- LCD Display
- Motors
- SPDT-Relay

Output Devices

LEDs

Speakers / Buzzers

Communications

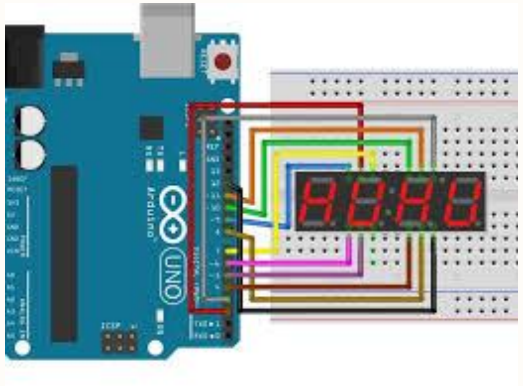
7-Segment Display

LCD Display

Motors

SPDT-Relay

7 segment displays



Output Devices

LEDs

Speakers / Buzzers

Communications

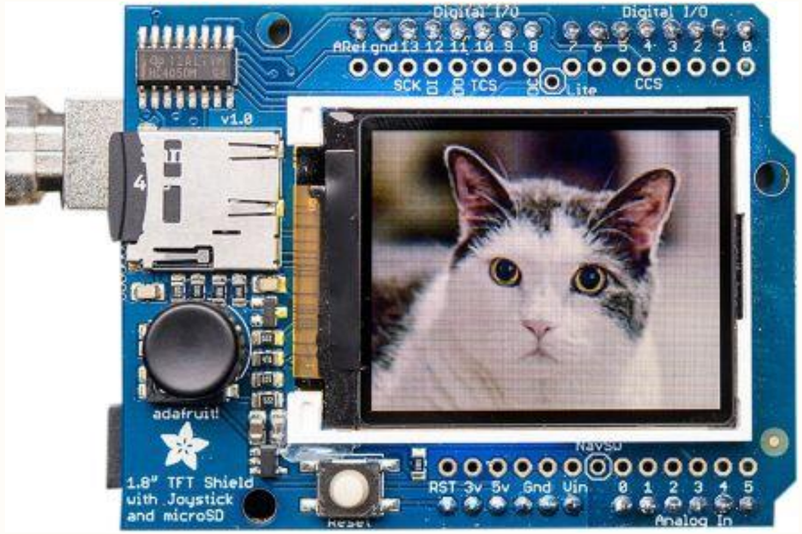
7-Segment Display

LCD Display

Motors

SPDT-Relay

LCD Displays



Output Devices

LEDs

Speakers / Buzzers

Communications

7-Segment Display

LCD Display

Motors

SPDT-Relay

Brushed DC Motor

Servo Motor

Brushed Coreless DC Motor

Stepper Motor

Output Devices

LEDs

Speakers / Buzzers

Communications

7-Segment Display

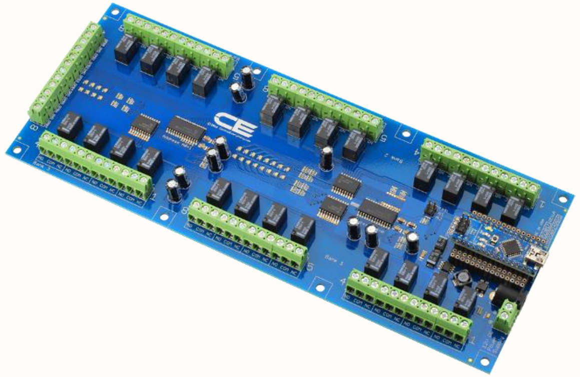
LCD Display

Motors

SPDT-Relay

SPDT Relay - 10 amps 120 – 240 volts

24 pack



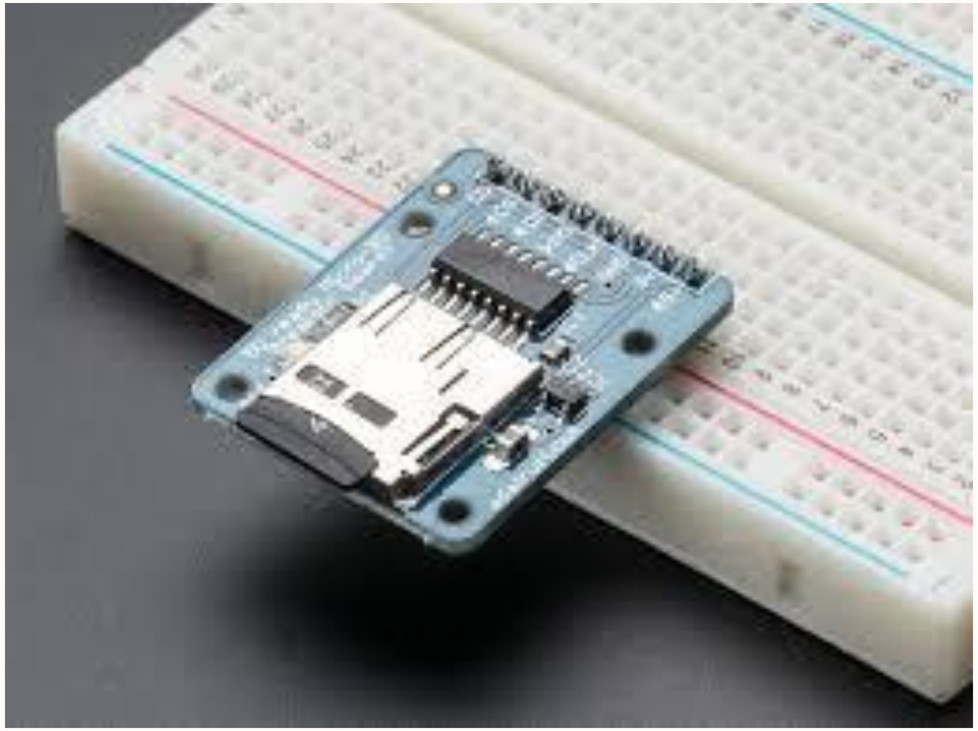
8 Pack

Requires a library.

Input / Output Devices

SD and MicroSD

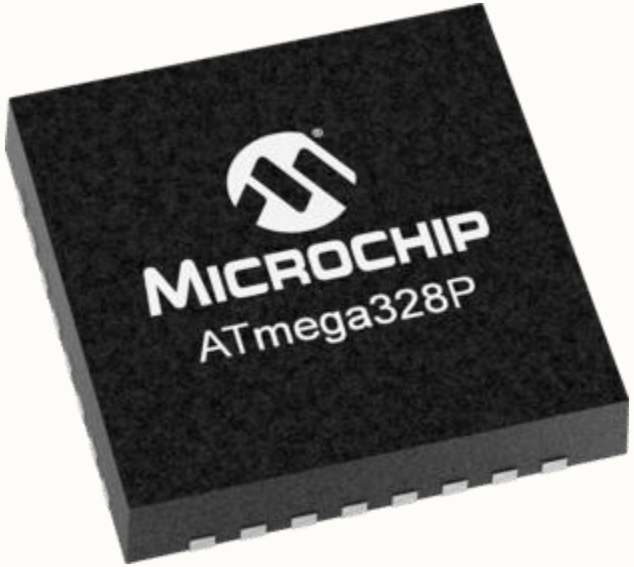
EEPROM



Input / Output Devices

SD and MicroSD

EEPROM



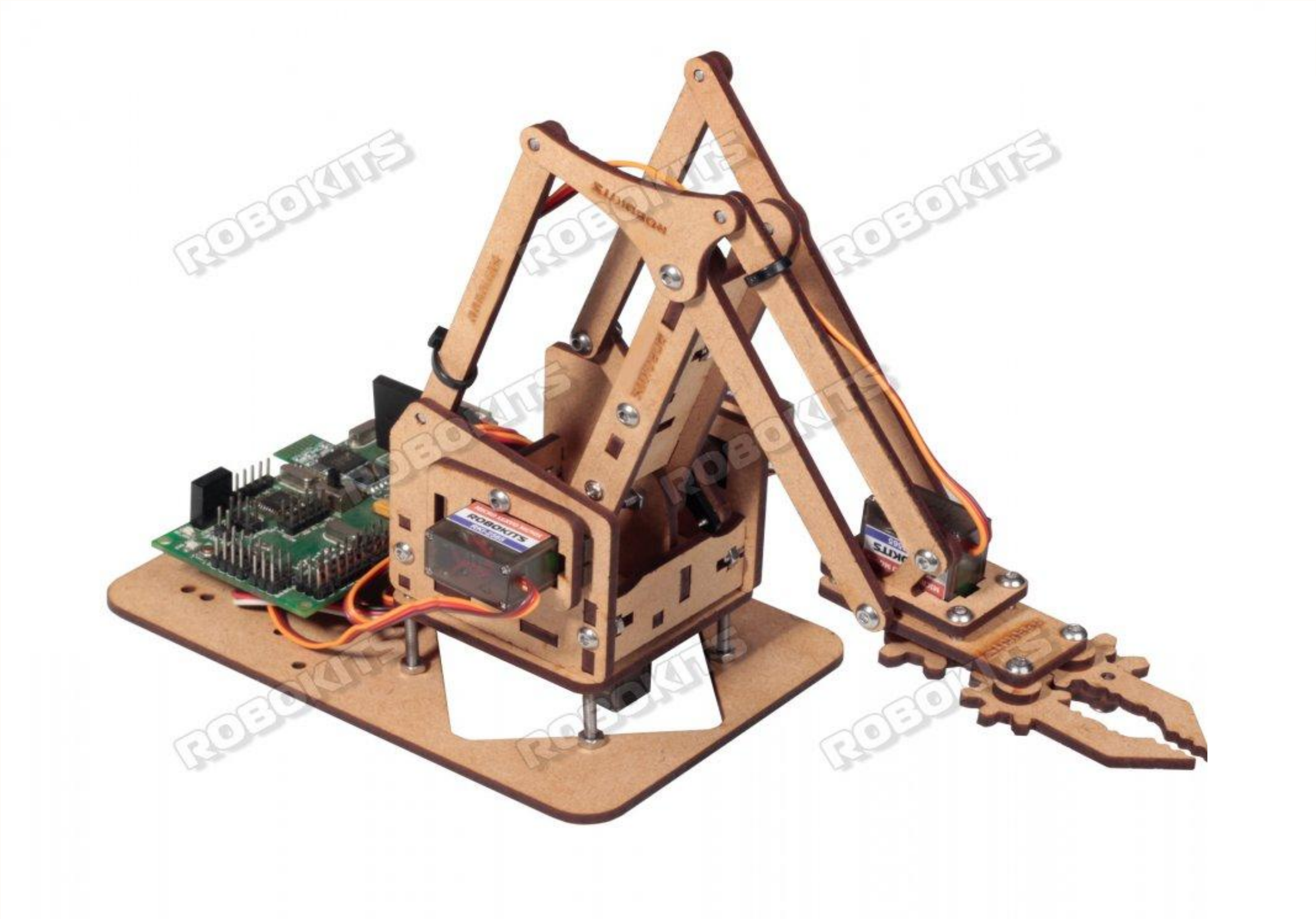
Electrically Erasable Programmable Read-Only Memory

Total lifetime of **~100,000 write cycles**

Project Ideas and Kits

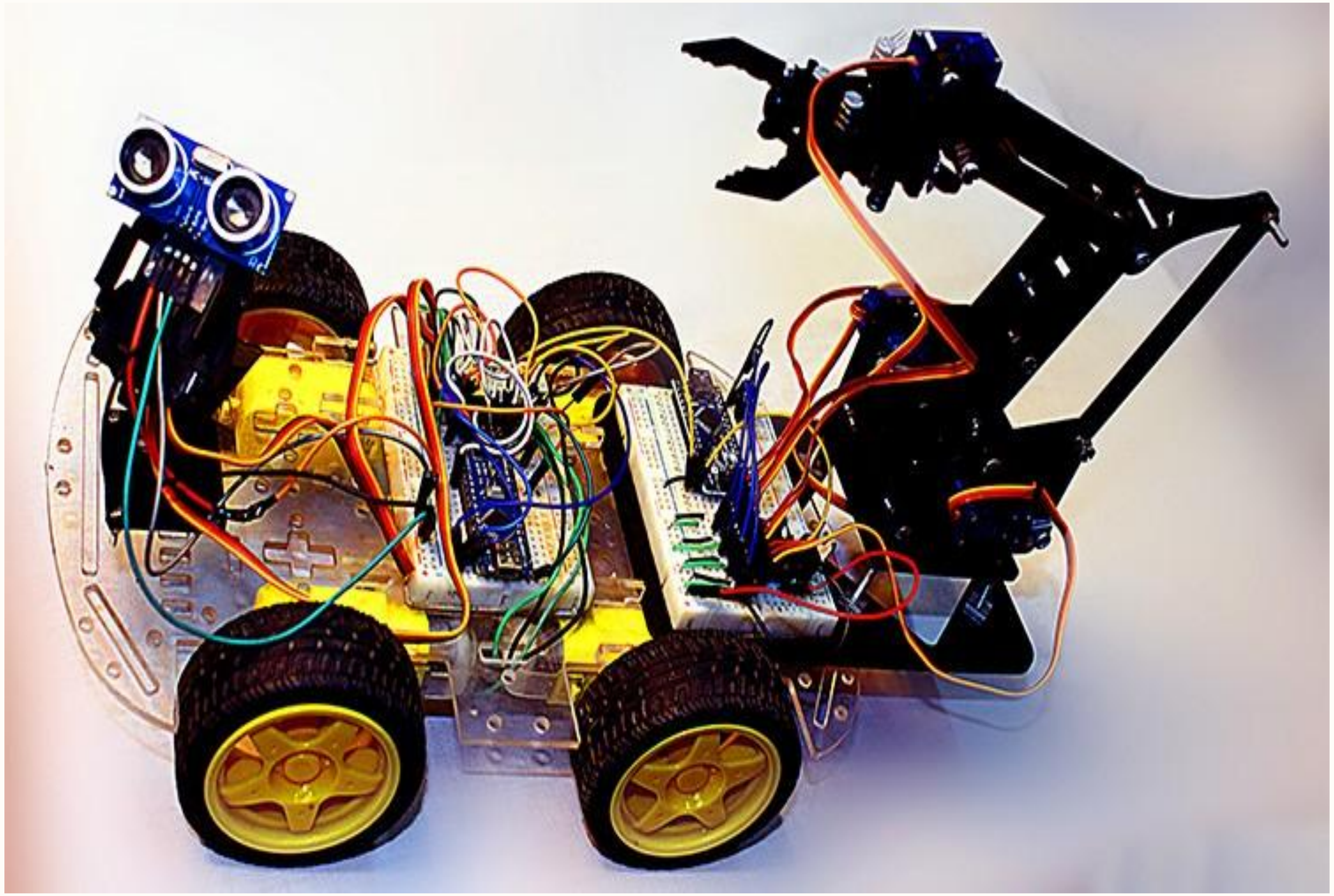
Project Kits

Need help loading and unloading?



Project Kits

Need to get to that un-reachable location?



Speedometer

Goals:

Measure and display scale train speed.

Facilitate speed matching for consists.

Tune reverse and forward speed.

Limit top speed to prototypical capabilities.

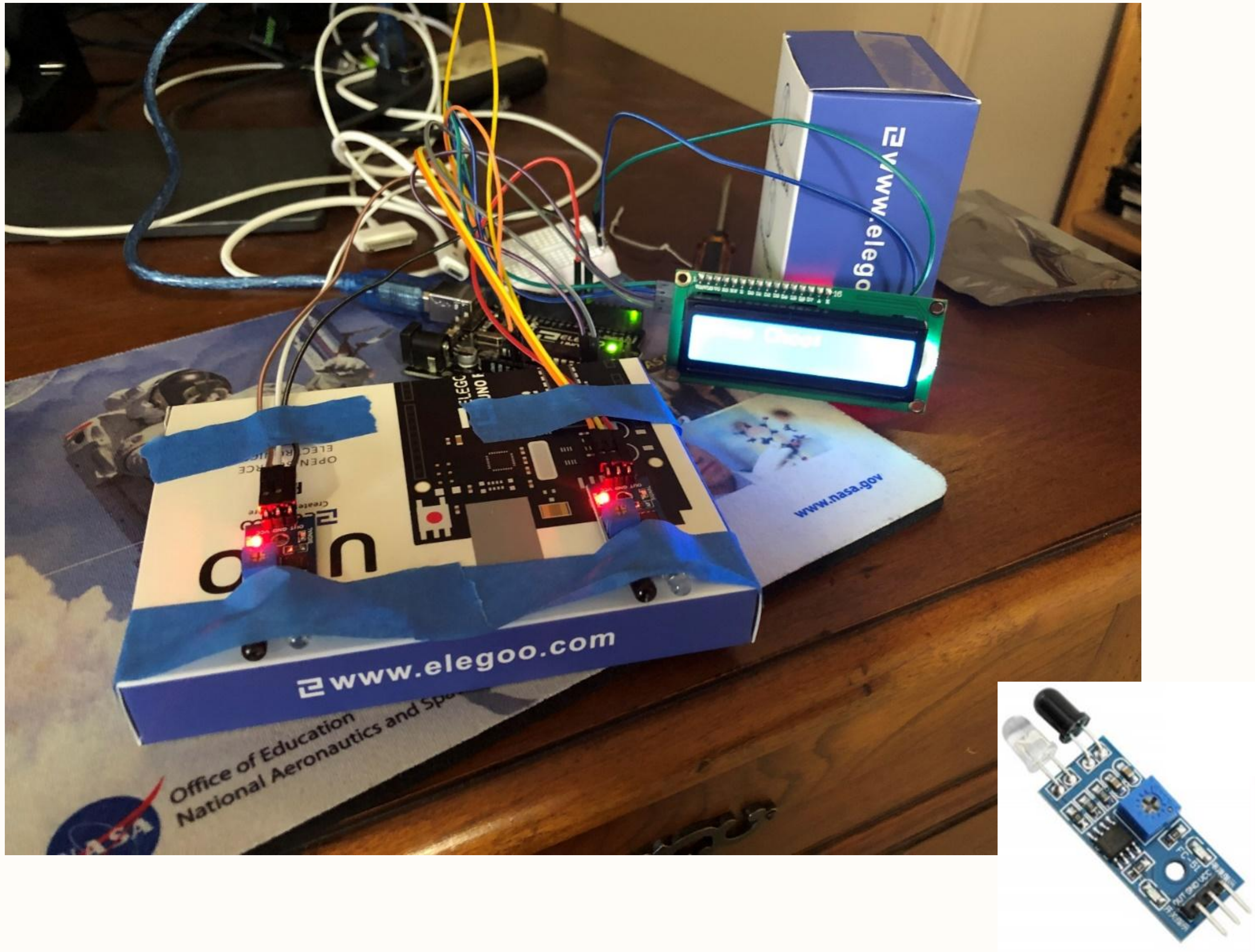
Perhaps make the train speed match the throttle indication.

Learn Arduino.

Speedometer

Iteration #1

It worked, proof of concept. Using IR sensors.

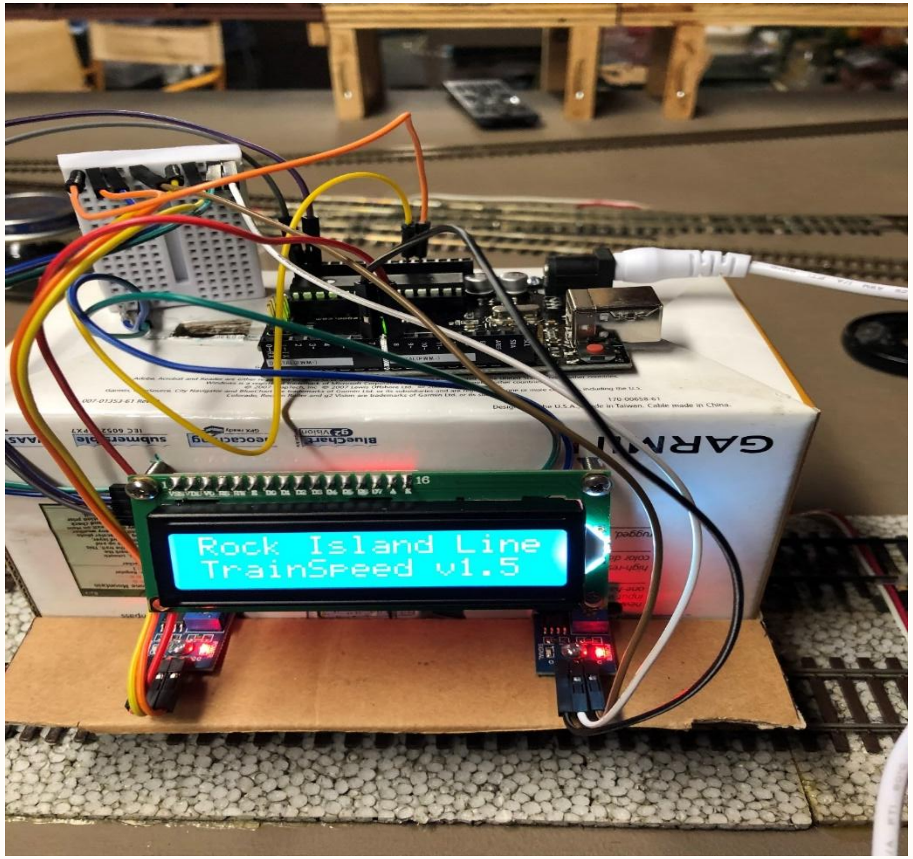


Speedometer

Iterations #2 & #3

Iteration #2 had the IR sensors on top pointing down. Sensitivity was poor due to varying car heights.

Iteration #3 moved the IR sensors to a horizontal position at wheel height. Consistent sensitivity but poor precision.



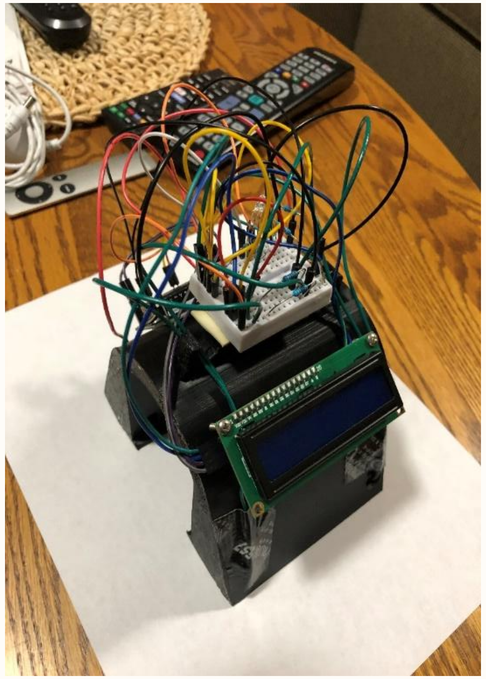
Speedometer

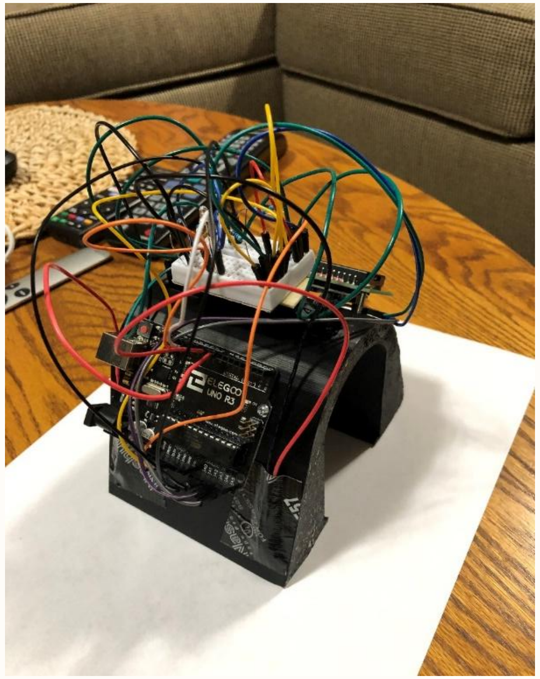
Iteration #4

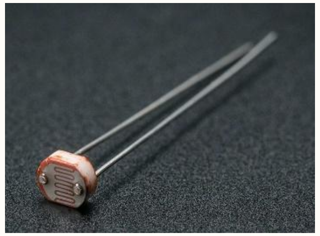
Switched to photocells, precision is very good. Photocells are an analog device whereas the IR sensors are digital.

Distance between the photocells is 3.1 inches. At .5 mph it takes 29 seconds to traverse the setup. An arbitrary reset occurs at 31 seconds. The LED on top turns red when measurement is in progress.

Both sensors need to be clear for 4 seconds before another measurement can be started. The LED on top turns green when ready.







Speedometer Coding

Following is a brief overview of the sketch, with the intent to show the parts of the program which include:

Includes
Constants
Setup()
Loop()

Direction detection.
Capture the elapsed time.
Calculate the speed.
Format the information for the LCD display.
Wait for a rest period.
Reset if all else fails.

Speedometer Code 1 of 6

```
#include <Wire.h>
#include <hd44780.h>
#include <hd44780ioClass/hd44780_I2Cexp.h>    // include i/o class header

//   To validate speed  goto   http://www.stonysmith.com/railroad/speedcalc.asp
// 30 seconds to transit 3" is .49 mph

const String greeting = "Rock Island Line";
const String version  = "v1.2";      // Maximum of 16 characters

const int left_LDR   = 1;      // Left LDR  - Light Dependent Resistor  pin # - Analog input
const int right_LDR = 2;      // Right LDR - Light Dependent Resistor  pin # - Analog input

const int ldrTrigger = 550;  // Dark ~ 1022   Lite ~ 356  approximately.

const int leftLED   = 8;        // LEDs pointing at the LDRs
const int rightLED = 9;

const int RED      = 12;         // Bi-Color LED Color
const int GREEN = 11;

// initialize the LCD
hd44780_I2Cexp lcd( 0x27, 16, 2 );   // set up the LCD's number of columns and rows:
```

Speedometer Code 2 of 6

```
const float scaleRatio      = 87.1;
const float r2lSeparation = 3.149606;  // Because of the alignment,  we need to handle both directions being different
const float l2rSeparation = 3.149606;  //   8.0 cm
const float mphRatio        = 1.467;       // To convert feet per second to miles per hour, divide by 1.467
float l2rScaleFeet;                                  // value is computed once
float r2lScaleFeet;                                  // value is computed once
float mph;

unsigned long  startTime;
unsigned long  endTime;

char bufMPH[8];   // buffer to hold MPH
char bufMS[8];     // buffer to hold transit time in milliseconds
String strMPH;

String topLine;
String bottomLine;

int direction;   //  either Left to Right   or   Right to Left
const int unknown = 0;
const int l2r      = 1;
const int r2l      = 2;
String lorr[] = { "Ukn", "L2R", "R2L" };                  //  a String representing the direction during Debug
const unsigned  long transitTimeout = 31000;  // Thirty One seconds, it takes ~30 seconds to transit 3" at .49 mph
const unsigned  long settleTime     = 4000;       // Time to wait before we take another measurement
unsigned long   settling;
unsigned long transitTime;            // in milliseconds
unsigned long transitExpire;          // This is when we give up on the pending transit
```

Speedometer Code 3 of 6

```
// ************************************************ Set up *********************************************
void setup() {        // put your setup code here, to run once:
 lcd.init();              // initialize the lcd
 lcd.backlight();    // lcd.noBacklight();

 pinMode( left_LDR, INPUT );
 pinMode( right_LDR, INPUT );

 pinMode( leftLED,  OUTPUT);      // LEDs shining on the LDRs
 pinMode( rightLED, OUTPUT);

 pinMode( LED_BUILTIN, OUTPUT);  // Orange LED
 pinMode( RED, OUTPUT);                // Red LED
 pinMode( GREEN, OUTPUT);           // Green LED

 lcd.setCursor(0, 0);   // First value is the Horizontal position index, second is the line index
 lcd.print( greeting );
 lcd.setCursor(0, 1);

 lcd.print("TrainSpeed " + version );
 clearCounters();

 l2rScaleFeet = ( l2rSeparation * scaleRatio) / 12 ;   // Calculate scale feet.  Should be 21.775 for HO
 r2lScaleFeet = ( r2lSeparation * scaleRatio) / 12 ;   // Calculate scale feet.  Should be 21.775 for HO

 digitalWrite(LED_BUILTIN, LOW);
 digitalWrite(GREEN, HIGH);

 digitalWrite( leftLED, HIGH);   // Turn on the LEDs
 digitalWrite( rightLED, HIGH);
 }
```

Speedometer Code 4 of 6

```
void loop() {  // put your main code here, to run repeatedly:

  if ( direction == r2l ) {
    if ( analogRead( left_LDR  ) > ldrTrigger )
      endTime = millis();

  } else if ( direction == l2r ) {
    if ( analogRead( right_LDR ) > ldrTrigger )
      endTime = millis();

  } else if ( direction == unknown )  {
    if ( analogRead( left_LDR ) > ldrTrigger ) {  // We have detected something
      startTime = millis();
      direction = l2r;
      bottomLine = topLine;
      lcd.setCursor(0, 1);  // col, row
      lcd.print( bottomLine );
      lcd.setCursor(0, 0);
      lcd.print(" Right to Left  ");
      transitExpire = startTime + transitTimeout;
      digitalWrite(LED_BUILTIN, LOW );
      digitalWrite(GREEN, LOW );
      digitalWrite(RED, HIGH );

    } else if ( analogRead( right_LDR ) > ldrTrigger ) {
        (  Same as above   )
    }
  }
```

Speedometer Code 5 of 6

```
if ( endTime > startTime ) {     // We have both time stamps,  compute the speed
  transitTime = endTime - startTime;      // transit time in ms.

  if ( direction == l2r )
   mph = (( l2rScaleFeet / transitTime) / mphRatio) * 1000;
  else
   mph = (( r2lScaleFeet / transitTime) / mphRatio) * 1000;

  dtostrf(mph, 5, 1, bufMPH );
  bufMPH[6] = 0;
  strMPH = bufMPH;
  sprintf( bufMS,  " %5lums", transitTime );
  topLine = strMPH +"mph"+ bufMS;
  lcd.setCursor(0, 0);
  lcd.print( topLine );

  clearCounters();
  settling = millis() + settleTime;

     // Wait until both LDRs are clear for 4 seconds
 while ( analogRead( left_LDR ) > ldrTrigger  || analogRead( right_LDR ) > ldrTrigger || settling > millis() ) {
   if ( analogRead( left_LDR ) > ldrTrigger || analogRead( right_LDR ) > ldrTrigger )
    settling = millis() + settleTime;
  }

 digitalWrite(LED_BUILTIN, HIGH );
 digitalWrite(RED, LOW );
 digitalWrite(GREEN, HIGH );
 }
```

Speedometer Code 6 of 6

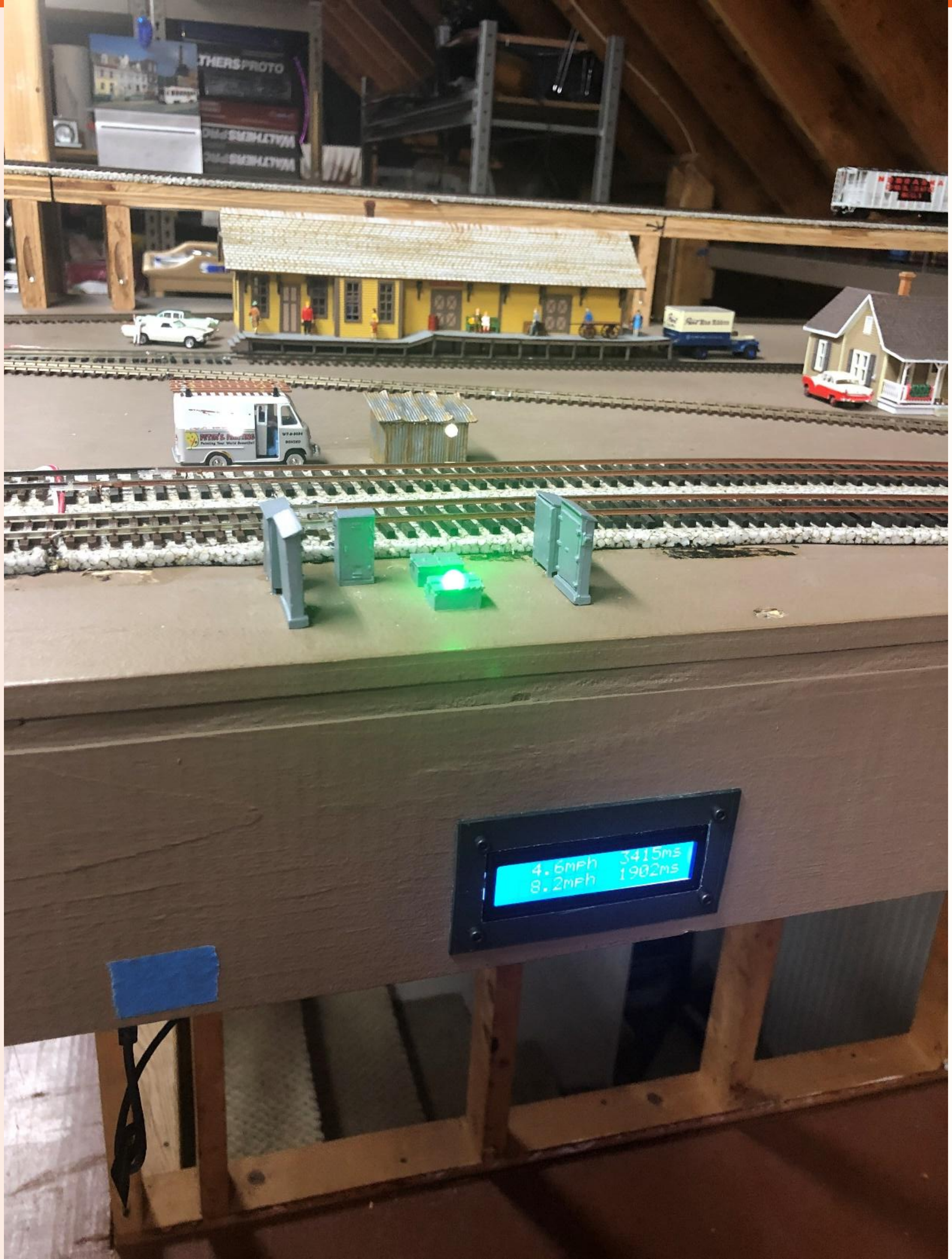
```
if ( millis() > transitExpire && direction > 0 ) {  // In case we never complete a transit
    lcd.setCursor(0, 0);                            // first pos = column index,  second pos is the row index
    lcd.print("Reset- Now Ready");
    clearCounters();
    digitalWrite(RED, LOW );
    digitalWrite(GREEN, HIGH );
  }
}     // End of loop


void clearCounters() {
  startTime = 0;
  endTime  = 0;
  direction  = unknown;
}
```

Speedometer Cost

Materials to build the Speedometer:

Item	Group Price	Price Each	Qty	Cost
Arduino UNO with USB cable	9.99	9.99	1	9.99
JANSANE 16x2 1602 LCD Display Screen	2 for 9.99	5.00	1	5.00
Breadboard	4 for 6.99	1.75	1	1.75
White LED		.19	2	.38
Bi-Color LED	100 / 6.79 - .07	.19	1	.19
Photocell	30 / 4.85 - .16	.95	2	1.90
Jumpers 120 pieces		6.98	?	6.98
USB Charger		5.99	1	5.99
IDE				Free
Total Cost				32.18

Installed Speedometer

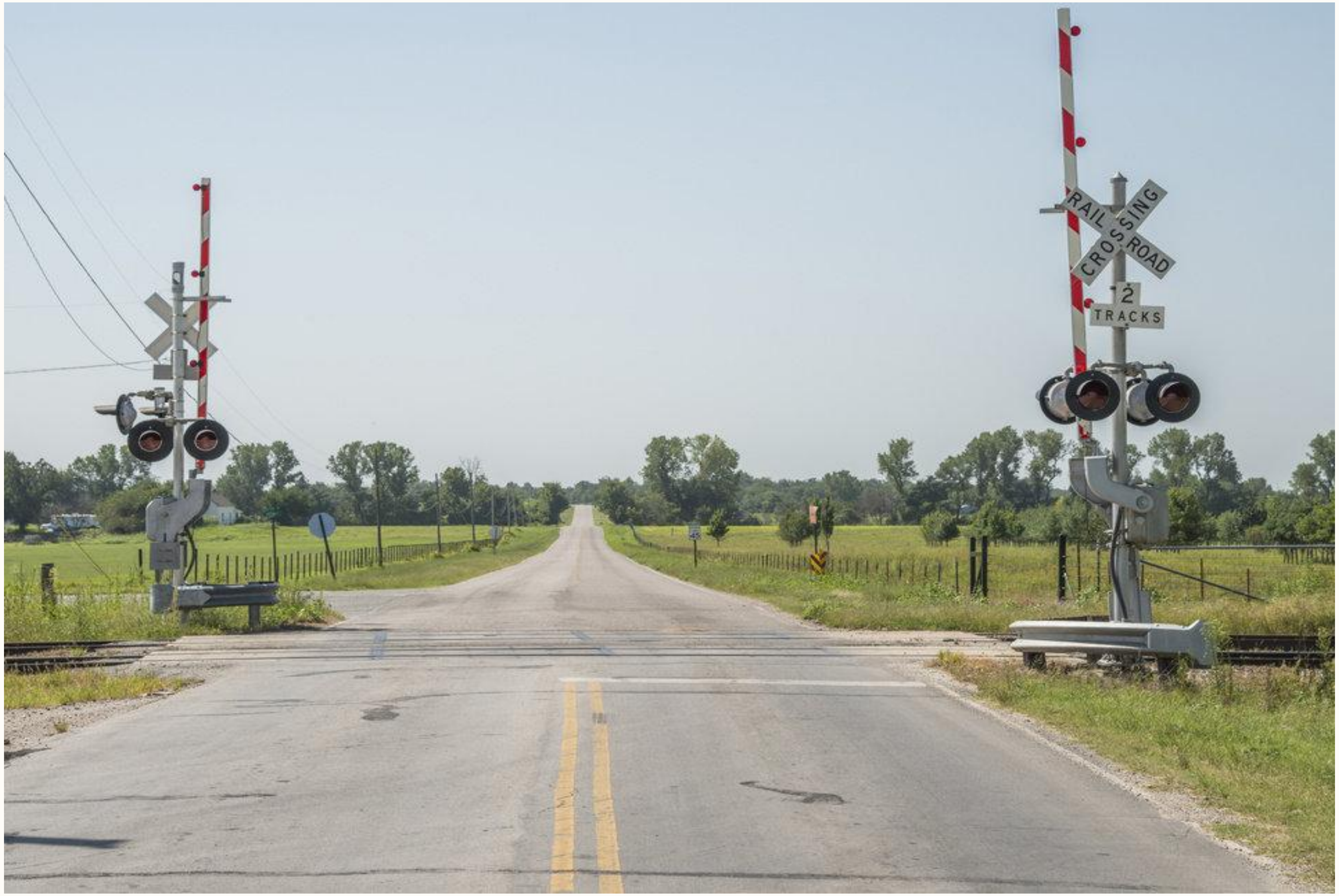


```
  4.6mph  3415ms
  8.2mph  1902ms
```

Now let’s consider another project:

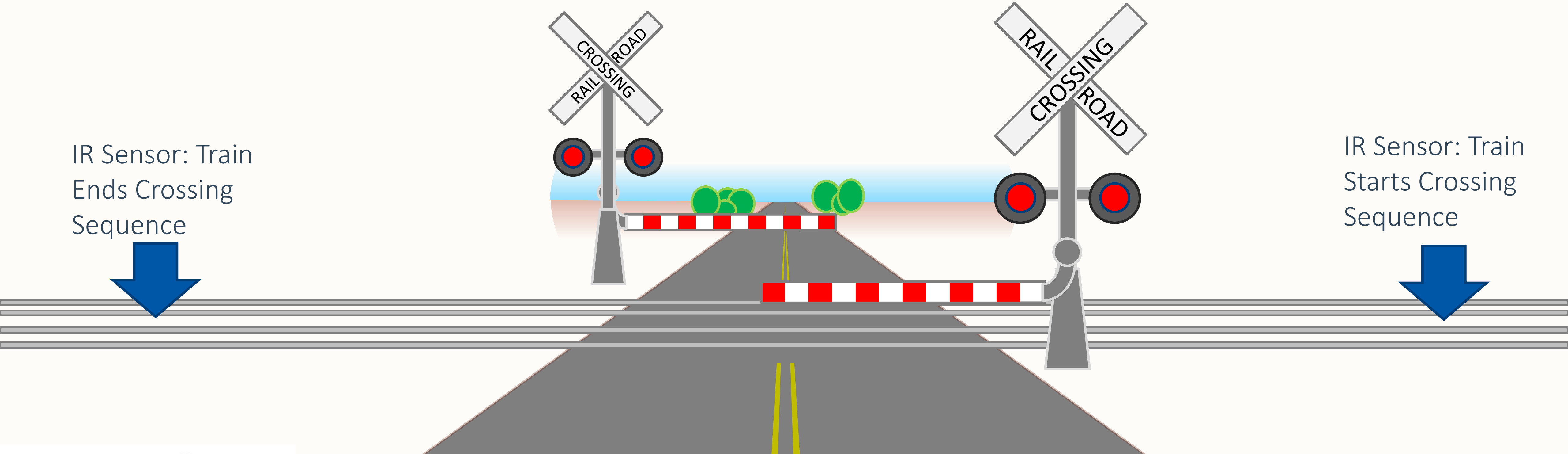
Grade Crossing

Grade Crossing



Lights and bell start before the arm comes down.

Grade Crossing



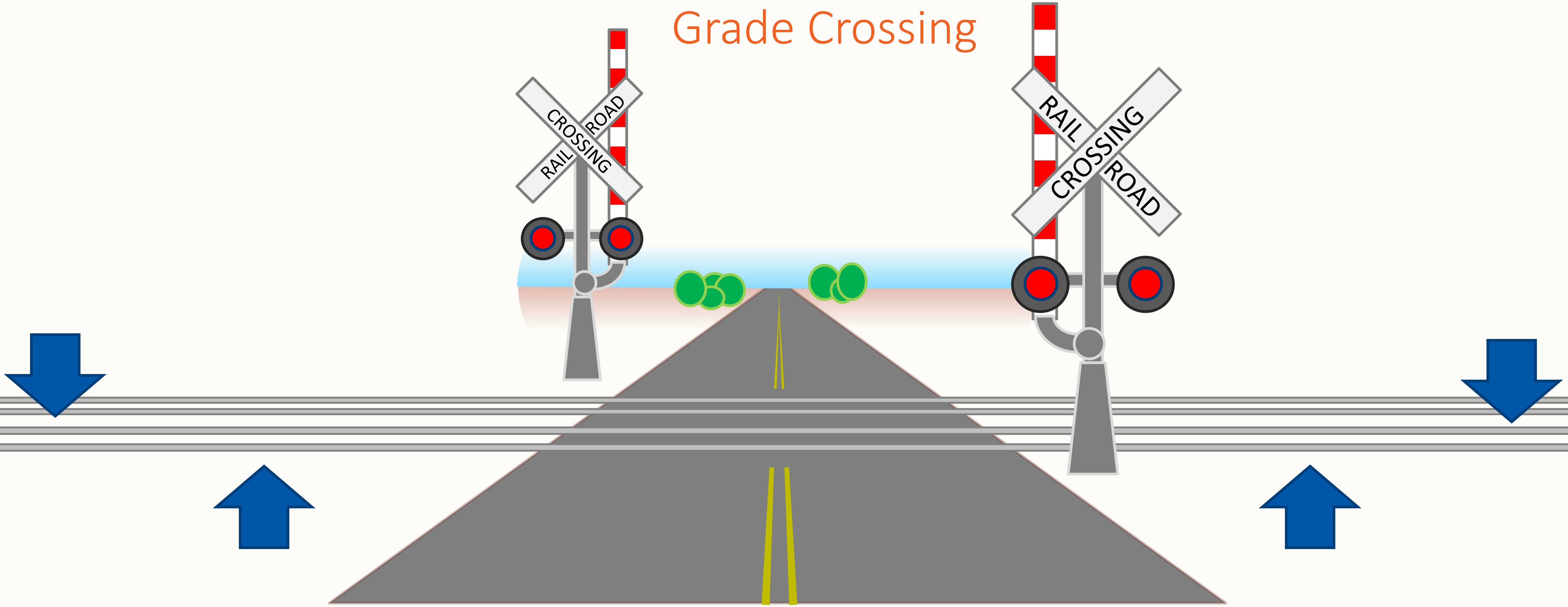
IR Sensor: Train Ends Crossing Sequence

IR Sensor: Train Starts Crossing Sequence

An IR sensor on either side of the road can start and stop the sequence.

What about the speed of the train?

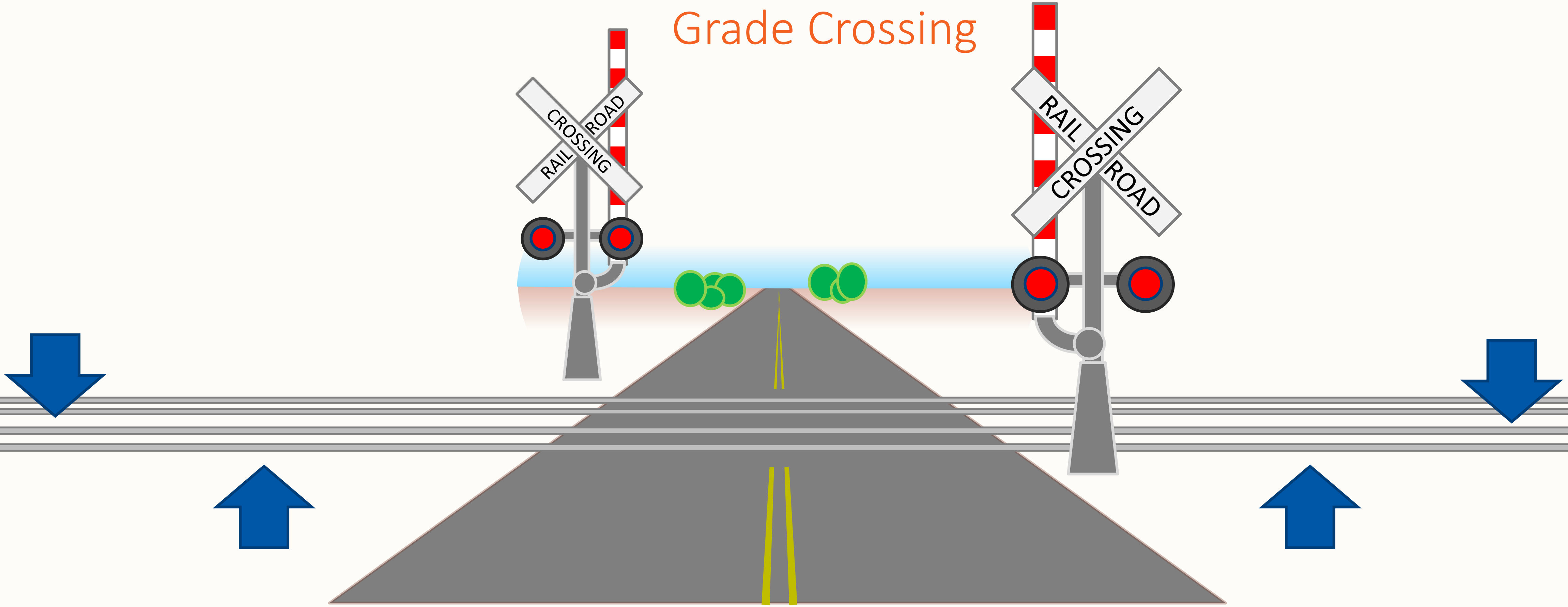
Grade Crossing



If we have two sensors on either side of the road, we can compute the speed and start the sequence.

We can also compute when to raise the gate.

Grade Crossing



What if we had three grade crossing?
Did ya notice that we have two tracks?
Perhaps we could use another method?

How to get Started?

Helps if you have a problem to solve.

Consider a Starter Kit.

Easy to do, but can be a bit challenging to get started.

Google for “how to” for wiring and code samples.

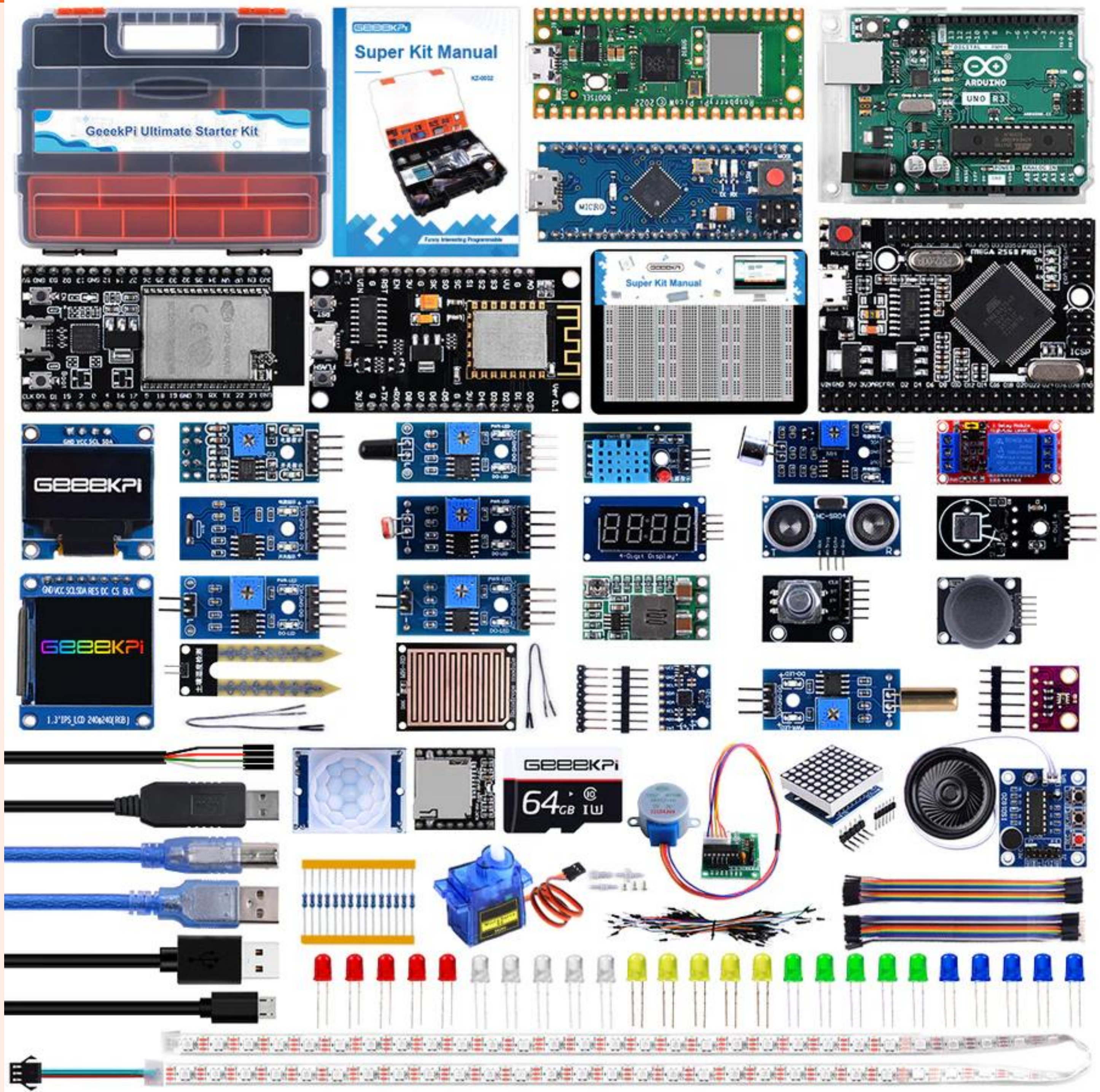
YouTube is another great resource.

Phone a friend.

Small “Learning C” clinics.

For more information…
Arduino.cc

Starter Kit


GeeekPi Ultimate Starter Kit
Super Kit Manual
ARDUINO
UNO R3
MICRO
Super Kit Manual
MEGA 2560 PRO
GeeekPi
1.3"IPS LCD 240x240(RGB)
HC-SR04
GeeekPi
64GB

ARDUINO™

Q & maybe A

Tonight….

- Review what is Arduino
    - Legally use of Arduino
    - Uses in model railroading
- Capabilities and devices available today
- Example Arduino project
- An example idea

Legal uses of Arduino

Arduino products are licensed under:

- GNU Lesser General Public License (LGPL)
- GNU General Public License (GPL)

Which permits the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form or as do-it-yourself (DIY) kits.

GNU and GNU/Linux is a product of the Free Software Foundation.

JMRI is distributed under the GNU license.

What is GNU

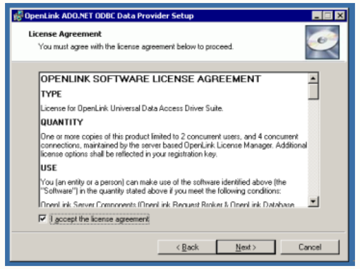
The wildebeest, also called the Gnu, is an antelope in the genus Connochaetes. It belongs to the family Bovidae, which includes antelopes, cattle, goats, sheep, and other even-toed horned ungulates.

Gnu is the name given to the project to create a Unix like operating system that is open and free for all to use.

The word “GNU” is a recursive acronym for “GNU's Not Unix!” 1983 Free Software Foundation (FSF)

FSF created the GPL and the LGPL to protect Open-Source software from being patented.

Hardware is typically not patentable, but a hardware specification is patentable.



OpenLink ADO.NET ODBC Data Provider Setup

License Agreement

You must agree with the license agreement below to proceed.

OPENLINK SOFTWARE LICENSE AGREEMENT

TYPE

License for OpenLink Universal Data Access Driver Suite.

QUANTITY

One or more copies of this product limited to 2 concurrent users, and 4 concurrent connections, maintained by the server based OpenLink License Manager. Additional license options shall be reflected in your registration key.

USE

You (an entity or a person) can make use of the software identified above (the "Software") in the quantity stated above if you meet the following conditions:

OpenLink Server Components (OpenLink Request Broker & OpenLink Database

☑ I accept the license agreement

Why Arduino

Significant changes of the past:

Barge traffic to Railroads
Transcontinental Railroad
Super Heating Steam
Steam to Diesel

From toy trains to scale models
Digital control of model locomotives
Sound in the model locomotives
Dead Rails

Arduino has great potential to enhance model railroading.

Bring intelligence to simple circuits.
Bring life to static art.
Assist in bringing the imagination to reality.

- This is a dupe - - cut one or the other

Every so often a change comes that seems to change everything

Looking in the past:

Barge traffic to Railroads
Transcontinental Railroad
Super Heating Steam
Steam to Diesel
Mobile phones

From toy trains to scale models
Digital control of model locomotives
Sound in the model locomotives
Dead Rails

Arduino has great potential to enhance model railroading.

Bring intelligence to simple circuits.
Bring life to static art.
Assist in bringing the imagination to reality.

Is Arduino production quality Hardware & Software?

Arduino is a general-purpose prototyping platform.

Arduino libraries are bloated in support of all the boards and functions.

There are no validation/testing requirements from the numerous manufactures or suppliers.

Production microcontrollers are much smaller, more specific and more cost effective.

Arduino can be, and is, used in commercial products.

GPL requires software libraries to be provided for when hardware is updated.

For modelers and tinkers, it is perfect.

A Post from Groups.io JMRI Users

Stefan Bartelski
Aug 6   #162539

Sorry Jim,
I was wrong :-(. I thought that it was working, but no. I could not get the programming side to work and the ops (throttle) was also not reliable. So I but the bullet and bought a motor shield from Arduino, $24 instead of $6. But now my DCC++ system does work 😁 .

So my $35 system became a $55 system (all

--
Stefan Bartelski

Home layout: The Blue Ridge Line, an HO representation of the L&N Etowah Old Line from Etowah to Elizabeth and the Marble Hill branch (Georgia Marble Railroad), set in 1986 (under construction)
Modular Layout: Shoofly module of the Country RRoads Modular group www.CountryRRoadModular.com