Arduino 102

Arduino for Model Railroading

Programming Sensors, Turnouts and Communication

Thomas Ose

What we will cover

- Quick Overview what we learned so far
- Function (reusable code)
- Introduction to Libraries
- Sensors
- Servos
- Relays
- Controlling Tortoise Switches
- Arduino to Arduino communications
- Q&A

What have we learned so far

- How to use the Arduino IDE
  - Edit a sketch
  - Upload a sketch to Arduino
  - Test a sketch
- Inputs
  - Digital (on/off)
  - Analogue (value)
  - Read
  - Write

What have we learned so far

- LED
  - How to turn them on
  - Resistor requirements
  - Blink
  - Fade
- Switches (push button)
- Sensors (light and temperature)

Function (reusable code)

- Functions are lines of code that perform a single reusable operation.
- Functions have to be defined before they are used
- Void function return no information
- Static Function have no variables

```
void turnOn()
{
        WriteDigital(pin1, High);
}

turnOn();
```

Function (reusable code)

- Static Function have no variable but can return values

```
boolean turnOn()
{
    WriteDigital(pin1, High);
    return true;
}

If (turnOn()) serialWrite(“pin 1 is on”);
```

Function (reusable code)

- Active Function have parameters that make it more flexible
- Active function may or may not return a value

```
void turnOn(int pinNumber)
{
	WriteDigital(pinNumber, High);
}

turnOn(1);
```

Function (reusable code)

- Active function may return a value

```
boolean turnOn(int pinNumber)
{
        WriteDigital(pinNumber, High);
        return true;
}

If (turnOn(1)) serialWrite("pin is on");
```

Function (reusable code)

- Parameters may be static as in previous examples
- Parameters can also be variable

```
boolean turnOn(int pinNumber)
{
        WriteDigital(pinNumber, High);
        return true;
}

int LEDpin = 1;
If (turnOn(LEDpin)) serialWrite(“pin ” + LEDpin +” is on”);
```

Function (reusable code)

- Return values can be Void for no return
- Return values can be of any type
- Return values can be used directly as any other variable
- Return values can also be used to define variables

Introduction to Libraries

- Are collections of functions that simplify re-usability
- They may reside in the source folder or in a library folder outside a project.
- Libraries must include a header file
  - Header files define the exposed functions and variables
- Libraries are included in a sketch with the #include statement that specifies the header file to include
- Libraries must be included before a function can be used
- Libraries can include other libraries.

Introduction to Libraries

- In the editor select

  Sketch –> Include Library –> (name of library)

- In the source code the following line will be added

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

Introduction to Libraries

- Some are just functional
- Exposing functions only
- Ease of reuse

```
pinMode(pin, type);
```

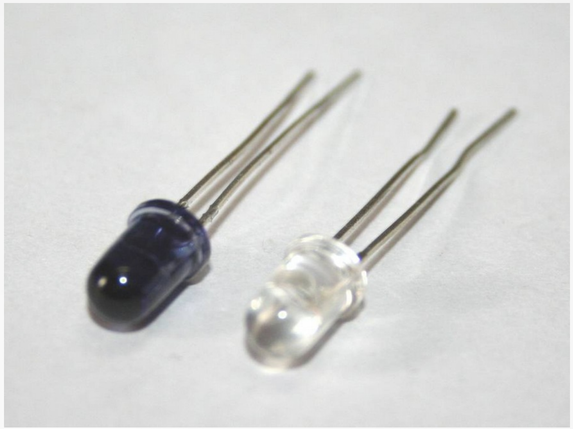
- Some are objects (class libraries)
- Derived objects inherit base functionality
- Can be extended

```
Servo MyServo;
MyServo.attach( pin);
```

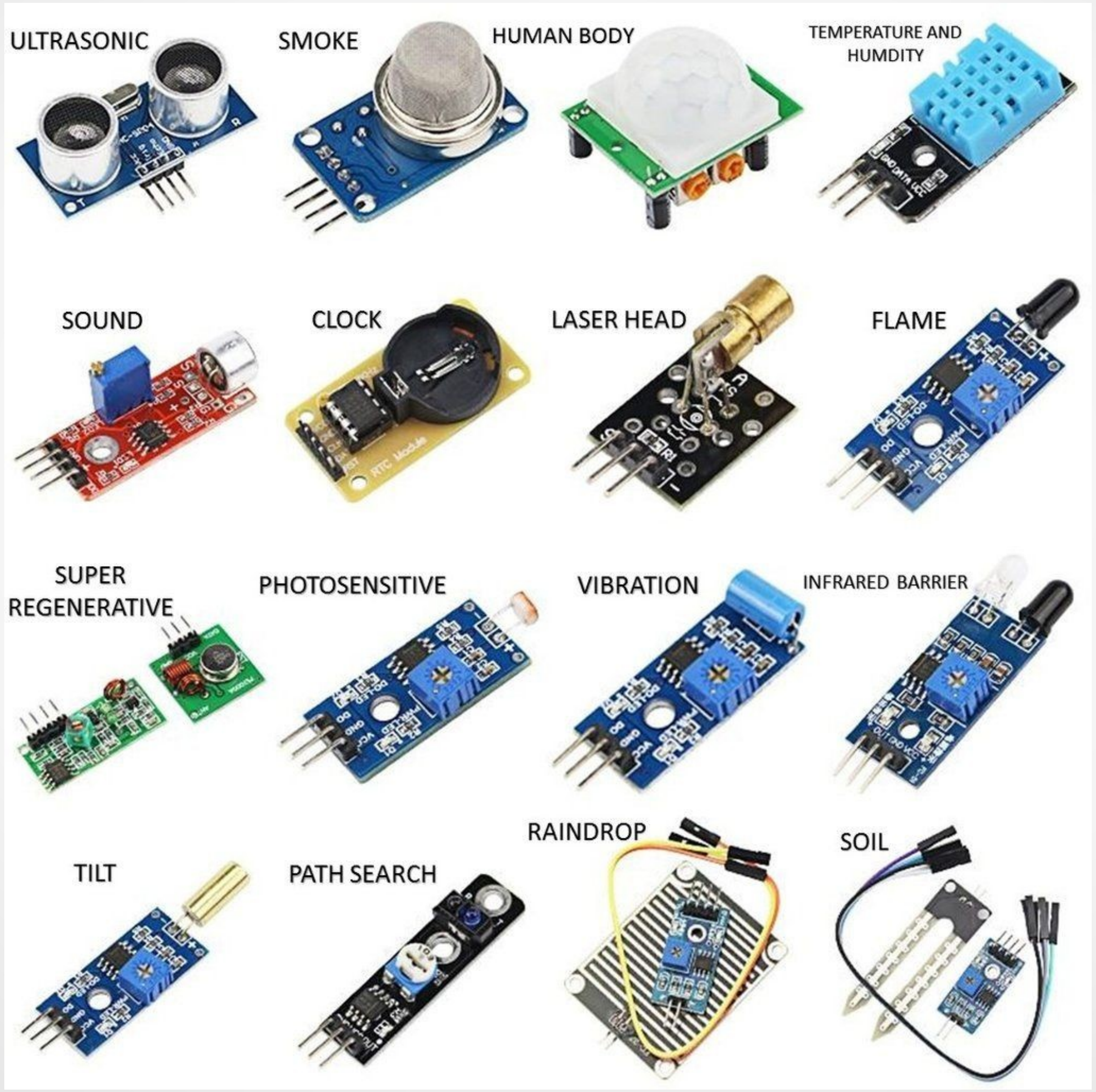
Sensors

- Sensors are the interface to the outside world
- Most sensors return an analogue value
- Some return a simple digital signal
- Sensors can be individual components
- Sensors can also be modules

Sensors components

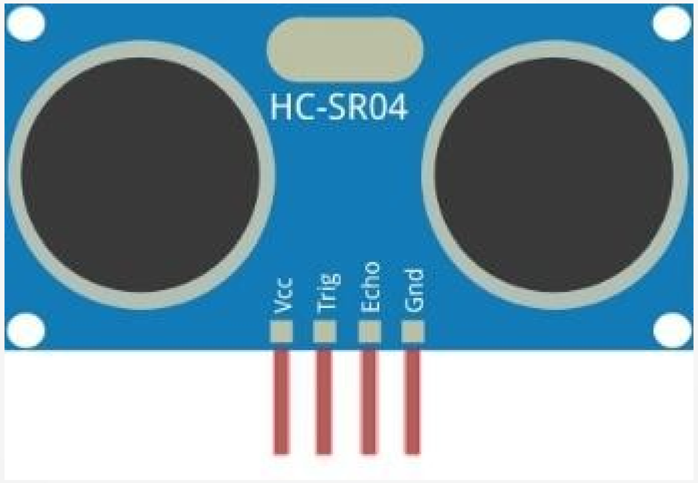


Sensors Modules



Tutorial – Ultrasonic Sensor

- Page 118 Project 14.1



What is a Servo

- A small device that incorporates a two wire DC motor, a gear train, a potentiometer, an integrated circuit, and an output shaft.
- Of the three wires that stick out from the motor casing, one is for power, one is for ground, and one is a control input line.
- The shaft of the servo can be positioned to specific angular positions by sending a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, then the angular position of the shaft changes

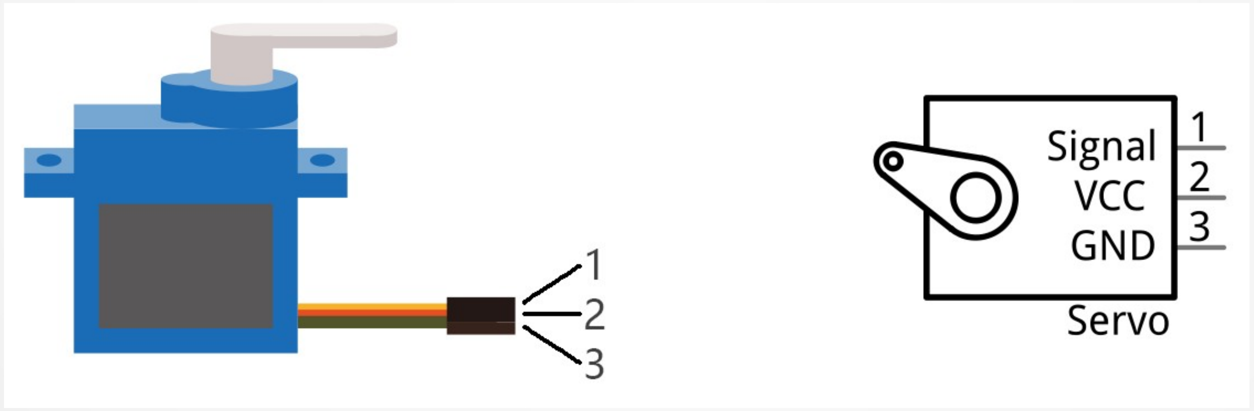
Reference: servocity.com

Servo Application

- Rotary or linear motion
- Used as switch machines
- Automation
- Crossing bar actuators
- Opening closing doors
- Will maintain position as long as signal does not change
  - Will seek position is left on
  - Recommend turning off if position is reached
- Does not require to be on at all times.

Tutorial - Servos

- Page 107 Project 11.2



Relays

- A relay is an electrically operated or electromechanical switch composed of an electromagnet, an armature, a spring and a set of electrical contacts.
- The electromagnetic switch is operated by a small electric current that turns a larger current on or off by either releasing or retracting the armature contact, thereby cutting or completing the circuit.
- Relays are necessary when there must be electrical isolation between controlled and control circuits, or when multiple circuits need to be controlled by a single signal.

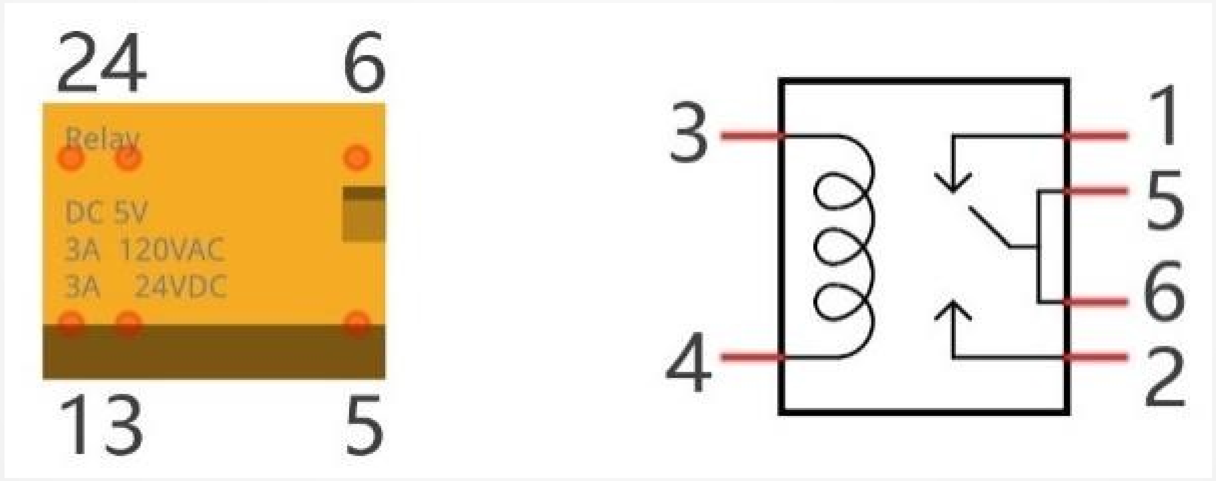
Reference: Techopedia.com

Relay applications

- Relays allow the control of larger voltage/current
- Relays isolate work load from control circuit
- Arduinos can drive relays to replace switches
- Powering staging tracks on/off
- Switching a track section between programming and ops mode.
- Control lighting
- Control turnout motors
- Turn automation on/off

Tutorial - Relays

- Page 90 Project 10

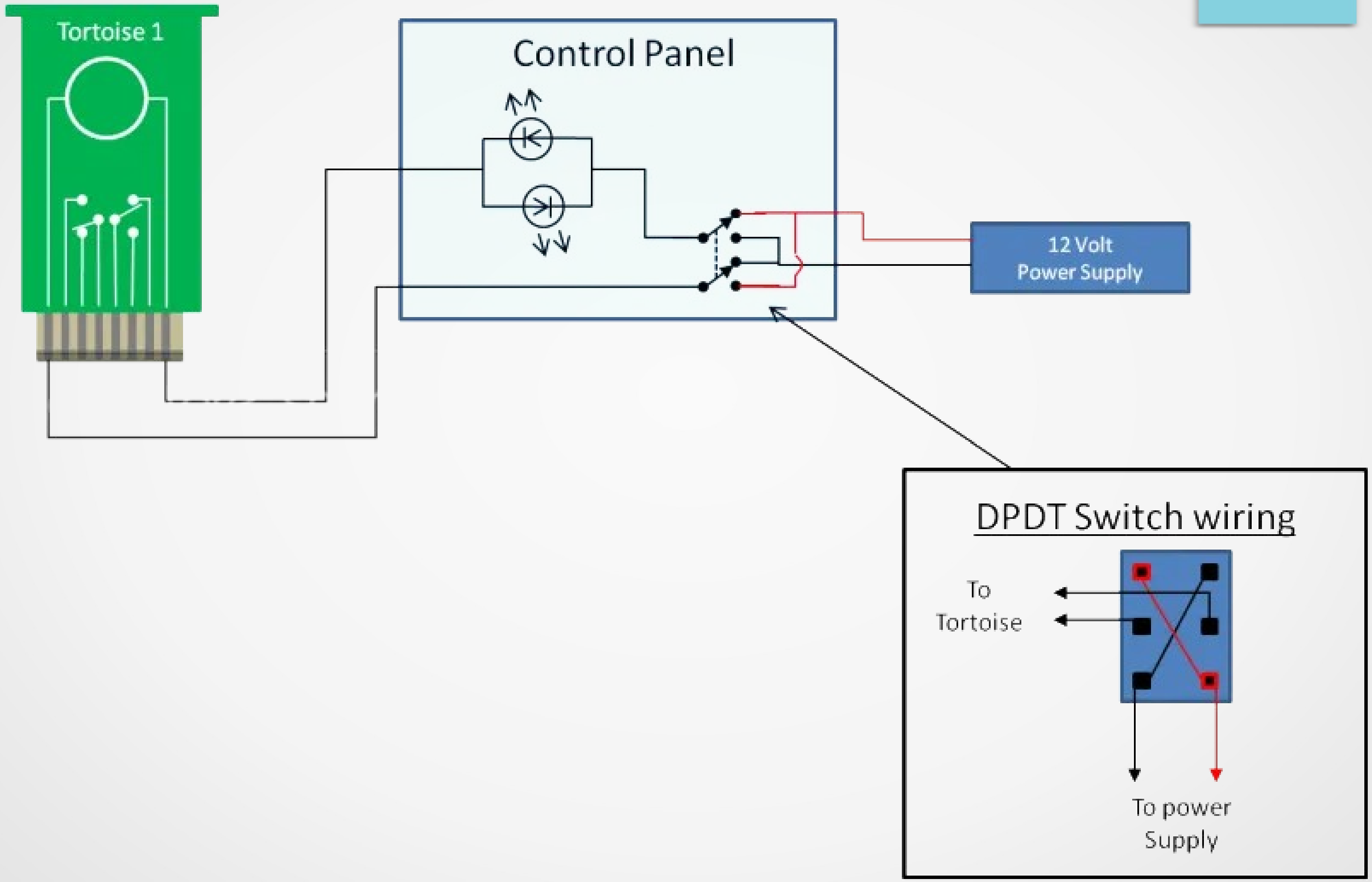


Controlling Tortoise Switches

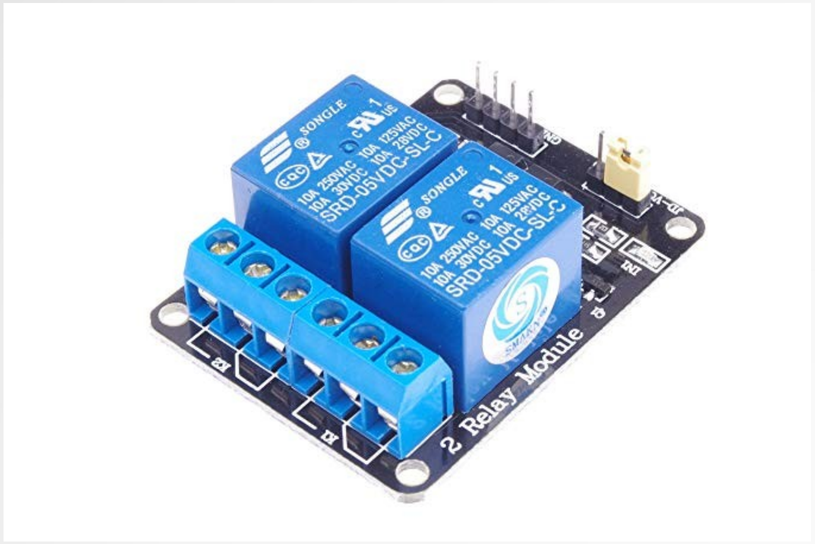
- Controls switch machines
- Referred to as a Stall Motor
- Power is always applied
- Works by reversing polarity
- Needs between 9 and 12 volts
- Typically operated by toggle switches
- Voltage to high for direct Arduino
- Can provide feedback for signals

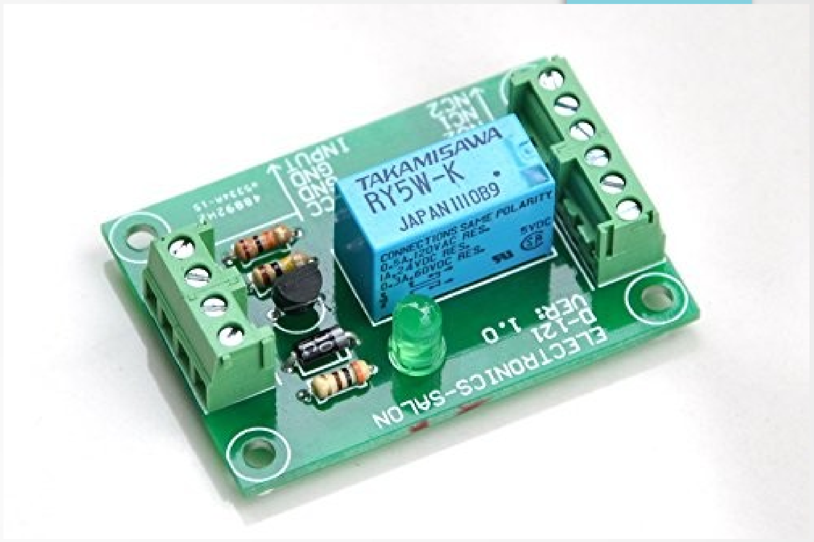
Circuitron
Tortoise™
Slow Motion Switch Machine
800-6000
WARNING: Warranty is void if label is damaged or removed!

Controlling Tortoise Switches



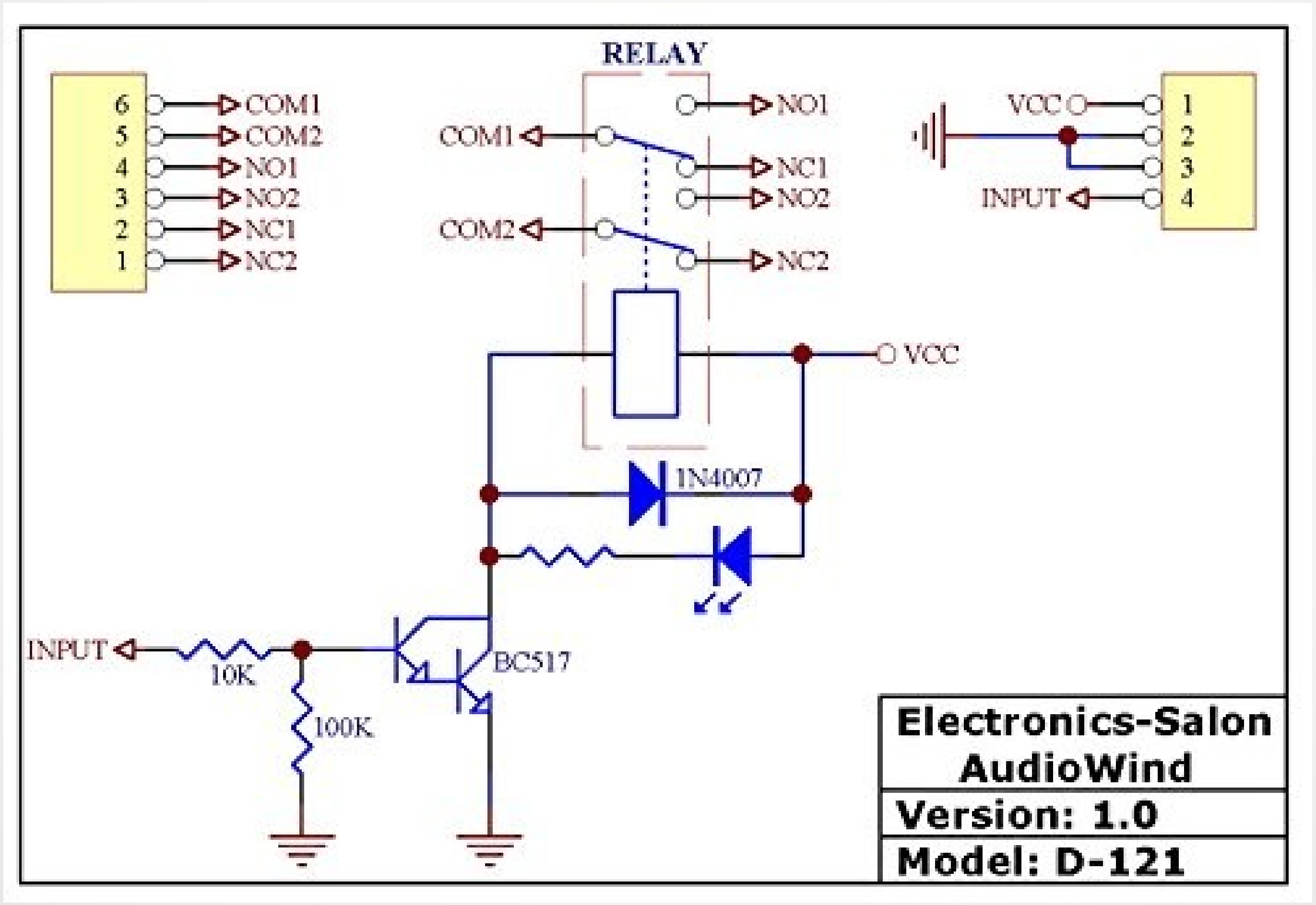
Controlling Tortoise Switches



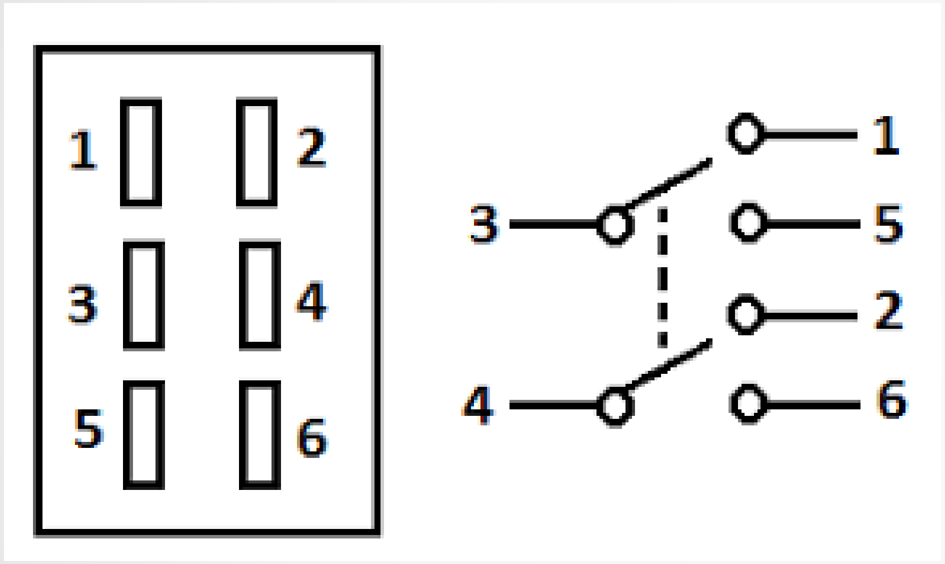


- Left side is a dual SPDT relay module
- Right side is a DPDT relay module
- Both will work but the DPDT only requires one output
- Programming is simpler with a DPDT

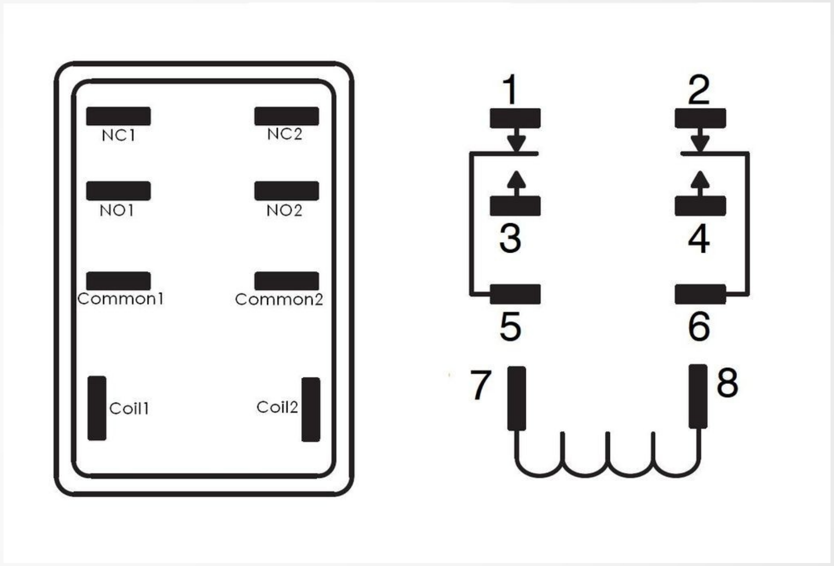
Controlling Tortoise Switches



Controlling Tortoise Switches

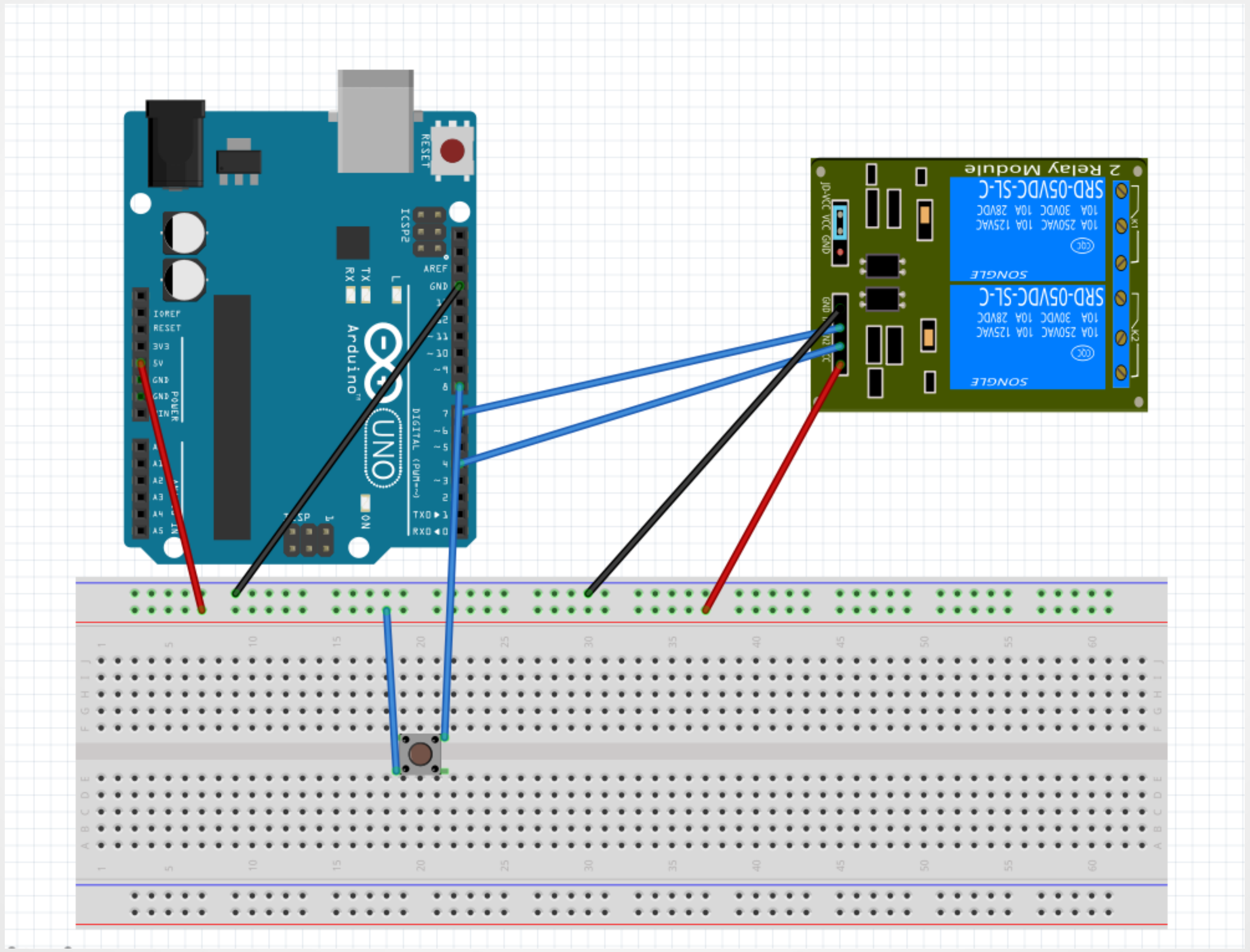


Toggle Switch



Relay

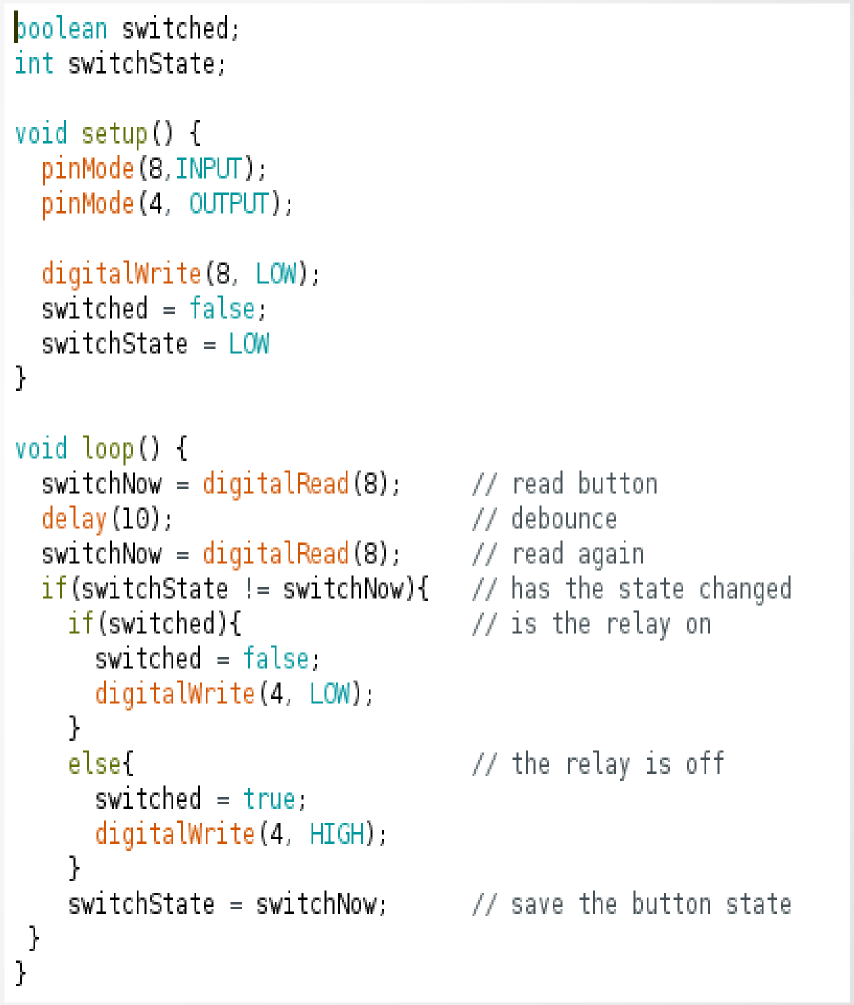
Controlling Tortoise Switches



Tortoise Code

- Define some state variables
- Set up the pins
- Set defaults
- Main loop

should add a delay at end otherwise will run every 10 micro seconds



```
boolean switched;
int switchState;

void setup() {
  pinMode(8,INPUT);
  pinMode(4, OUTPUT);

  digitalWrite(8, LOW);
  switched = false;
  switchState = LOW
}

void loop() {
  switchNow = digitalRead(8);       // read button
  delay(10);                        // debounce
  switchNow = digitalRead(8);       // read again
  if(switchState != switchNow){     // has the state changed
    if(switched){                   // is the relay on
      switched = false;
      digitalWrite(4, LOW);
    }
    else{                           // the relay is off
      switched = true;
      digitalWrite(4, HIGH);
    }
    switchState = switchNow;        // save the button state
 }
}
```

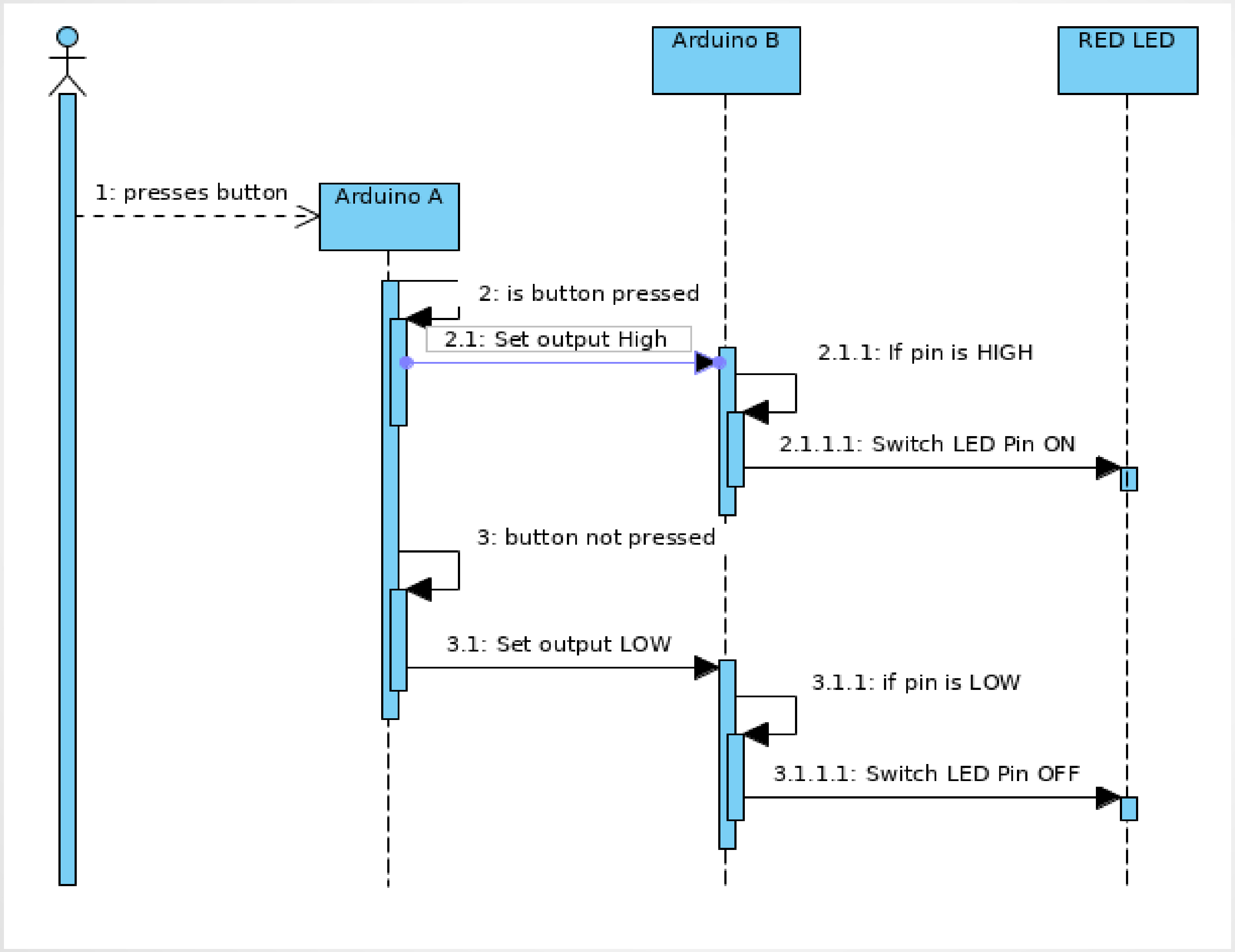
Other thoughts on Tortoise

- Example only deals with throwing switch
- Additional option
  - Add return from Tortoise to confirm position
  - Add LED output to Arduino for indication
  - Add Signal logic to Arduino to drive Signal head
  - Add NMRADCC library to add DCC funtionality
    - DCC control of turnout
    - JMRI interface to turnout

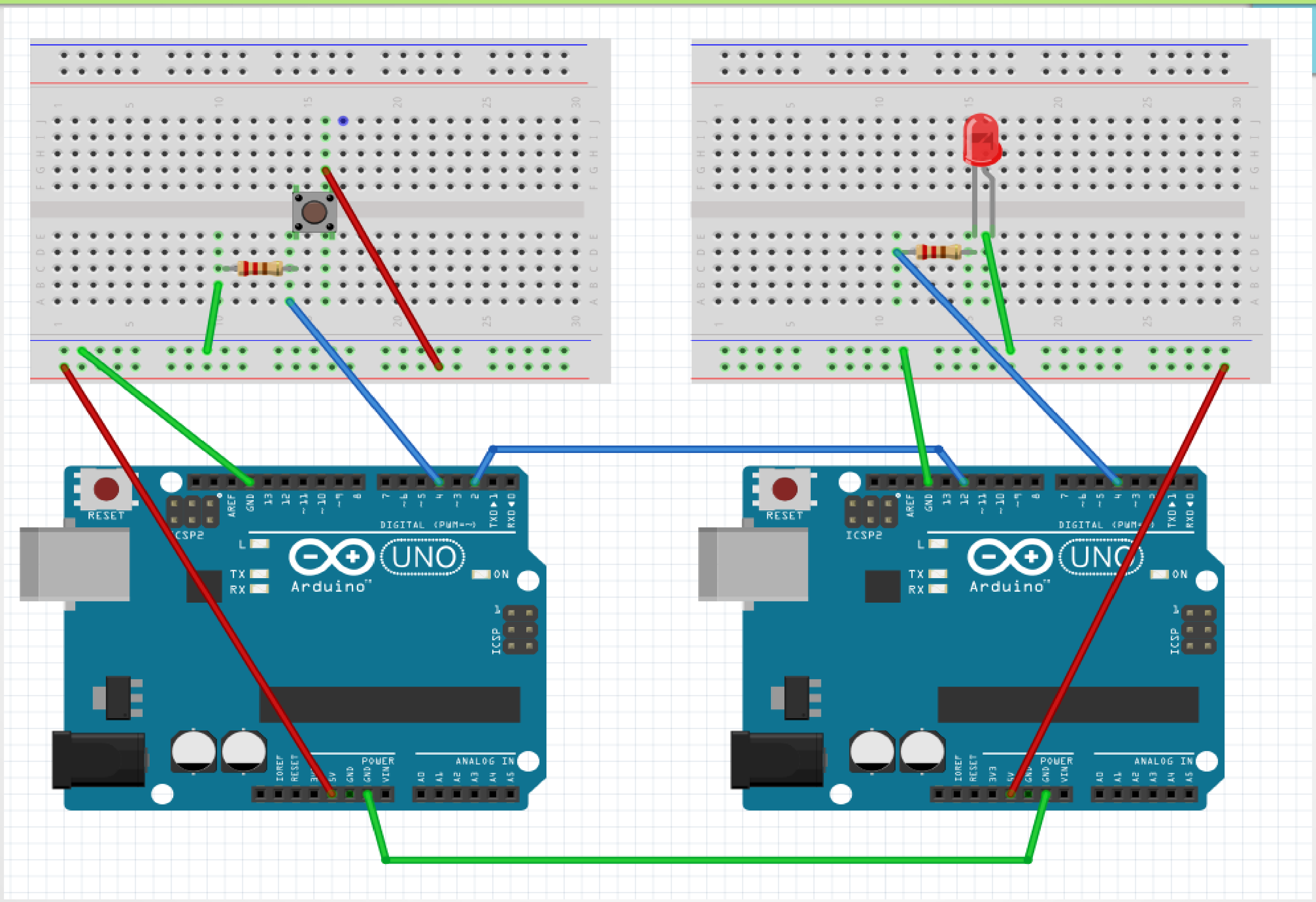
Arduino to Arduino communications

- Arduino can communicate among themselves
- Communication can be one to one
- Communication can also be one to many
- Communication can be one or two way
- Simple one-way point to point using pins
- Serial communication is more complex and versatile
  - One or two way
  - Message driven
  - Multiple device capable
- Arduino can communicate via SPI or network

Simple point to point



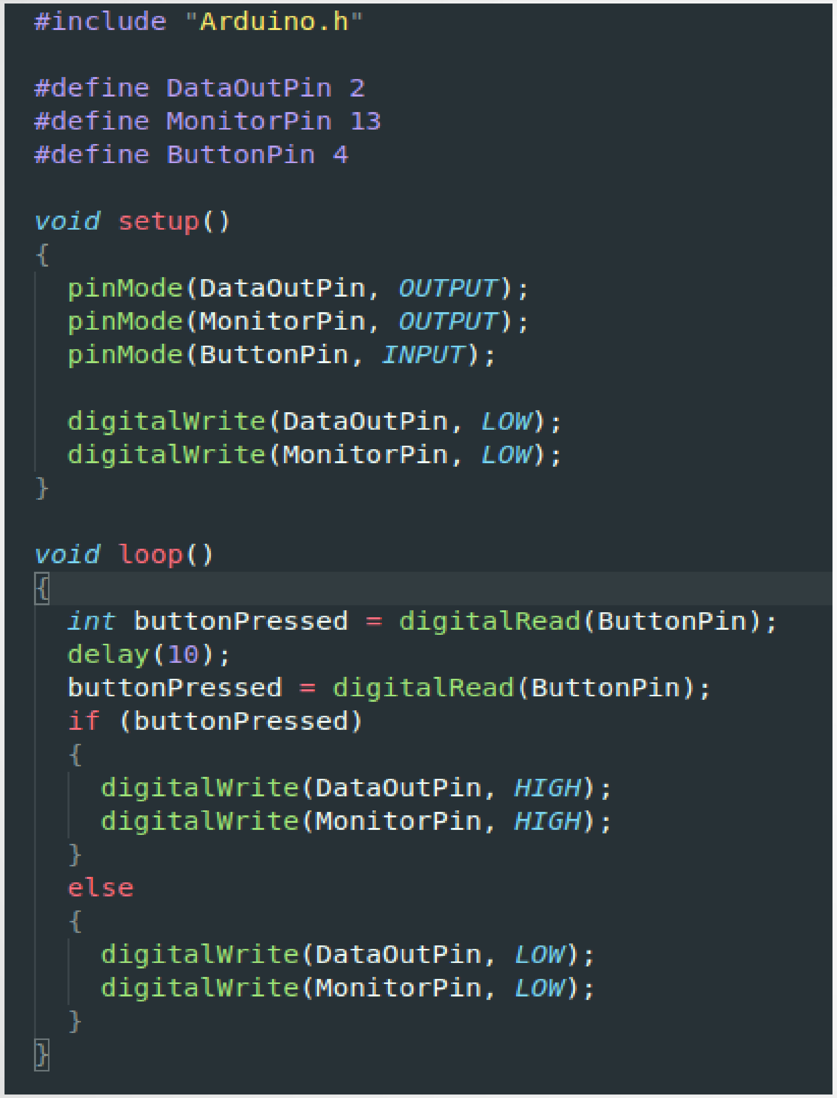
Simple point to point - wiring



Master

Slave

Simple point to point - Code



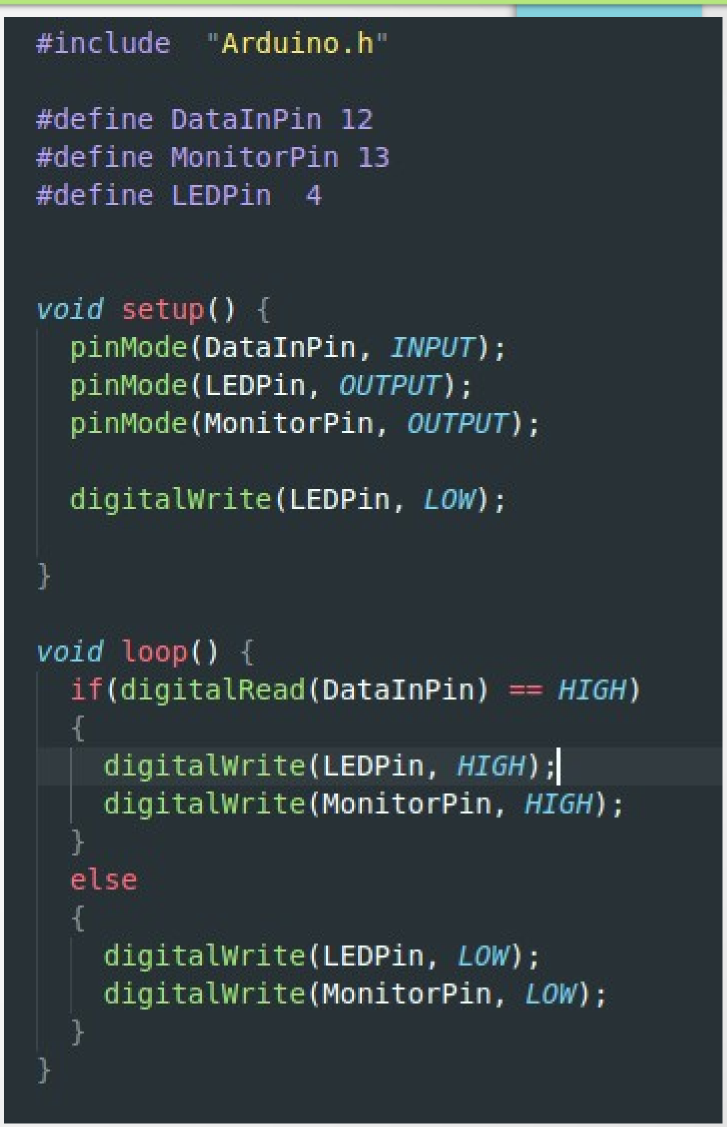
```
#include "Arduino.h"

#define DataOutPin 2
#define MonitorPin 13
#define ButtonPin 4

void setup()
{
  pinMode(DataOutPin, OUTPUT);
  pinMode(MonitorPin, OUTPUT);
  pinMode(ButtonPin, INPUT);

  digitalWrite(DataOutPin, LOW);
  digitalWrite(MonitorPin, LOW);
}

void loop()
{
  int buttonPressed = digitalRead(ButtonPin);
  delay(10);
  buttonPressed = digitalRead(ButtonPin);
  if (buttonPressed)
  {
    digitalWrite(DataOutPin, HIGH);
    digitalWrite(MonitorPin, HIGH);
  }
  else
  {
    digitalWrite(DataOutPin, LOW);
    digitalWrite(MonitorPin, LOW);
  }
}
```



```
#include  "Arduino.h"

#define DataInPin 12
#define MonitorPin 13
#define LEDPin  4


void setup() {
  pinMode(DataInPin, INPUT);
  pinMode(LEDPin, OUTPUT);
  pinMode(MonitorPin, OUTPUT);

  digitalWrite(LEDPin, LOW);

}

void loop() {
  if(digitalRead(DataInPin) == HIGH)
  {
    digitalWrite(LEDPin, HIGH);
    digitalWrite(MonitorPin, HIGH);
  }
  else
  {
    digitalWrite(LEDPin, LOW);
    digitalWrite(MonitorPin, LOW);
  }
}
```

Other thoughts on Communication

- Example is very simple and limited
- Switch communication to serial interface
  - Use AltSoftSerial library
  - Able to send different conditions
  - Create a packet and talk to multiple Arduinos
    - Add an address that the slaves recognizes
    - Add a data packet for instruction
  - Good model for switch communication
- Other protocols possible, but much more complicated

Q & A

Thank You for your time

Thomas Ose   tmo@osemicro.com

Files are found at
https://www.omsremodels.com