Approach Indicator

Revision v 0.5 02/07/2022

Seth Neumann, seth@modelrailroadcontrolsystems.com
Jon Schmidt, jontenor@gmail.com

Introduction

This document describes the Approach Indicator board and how to assemble and install it.

Revision History

V0.5 - February 7, 2022 SCN
V0.4 –October 05, 2021 JES
V0.3 –October 05, 2021 JES
V0.2 –September 21, 2021 JES
V0.1 – first pass – September 15, 2021

Table of Contents

Table of Figures

Table of Tables

1 INTRODUCTION

This board alerts a train order operator that a train is on approach ("on the bell") and that he should contact the Dispatcher to determine if there are additional orders for the train. It can handle up to 4 stations, 2 with 2 sets of 2 photo transistors or photo resistors and 2 more with a logic output from a detector such as our Enhanced Optical Position Detector. The sensors are paired in each approach direction so that unwanted "dinging" does not occur when a train leaves the station.

Features:

- Supports one or two stations with internal conditioning of photo sensor signals
- Supports an additional one or two stations with digital inputs from detector assemblies
- All connections including +5 and ground are brought to 0.100 connectors
- A 2.1mm barrel jack is provided if you choose to supply power from a 5V wall wart.
- All components use through-hole technology for ease of assembly and repair.
- Logic level outputs are provided for each station, as well as isolated "dry contacts" on relays. The relays are rated at 1A @ 48VDC.
- The relay outputs can optionally connect to ground by use of onboard jumpers
- Each output has an associated LED which is helpful in debugging, when relays are equipped

This board was designed by Seth Neumann, the code was developed by Jon Schmidt.

Schematic, circuit board layouts, CAD files and code are available on the product page on our website and on my GitHub page https://github.com/SethNeumann/MRCS-Approach-Indicator

2 IDENTIFICATION AND INFORMATION

2.1. BOARD LAYOUT

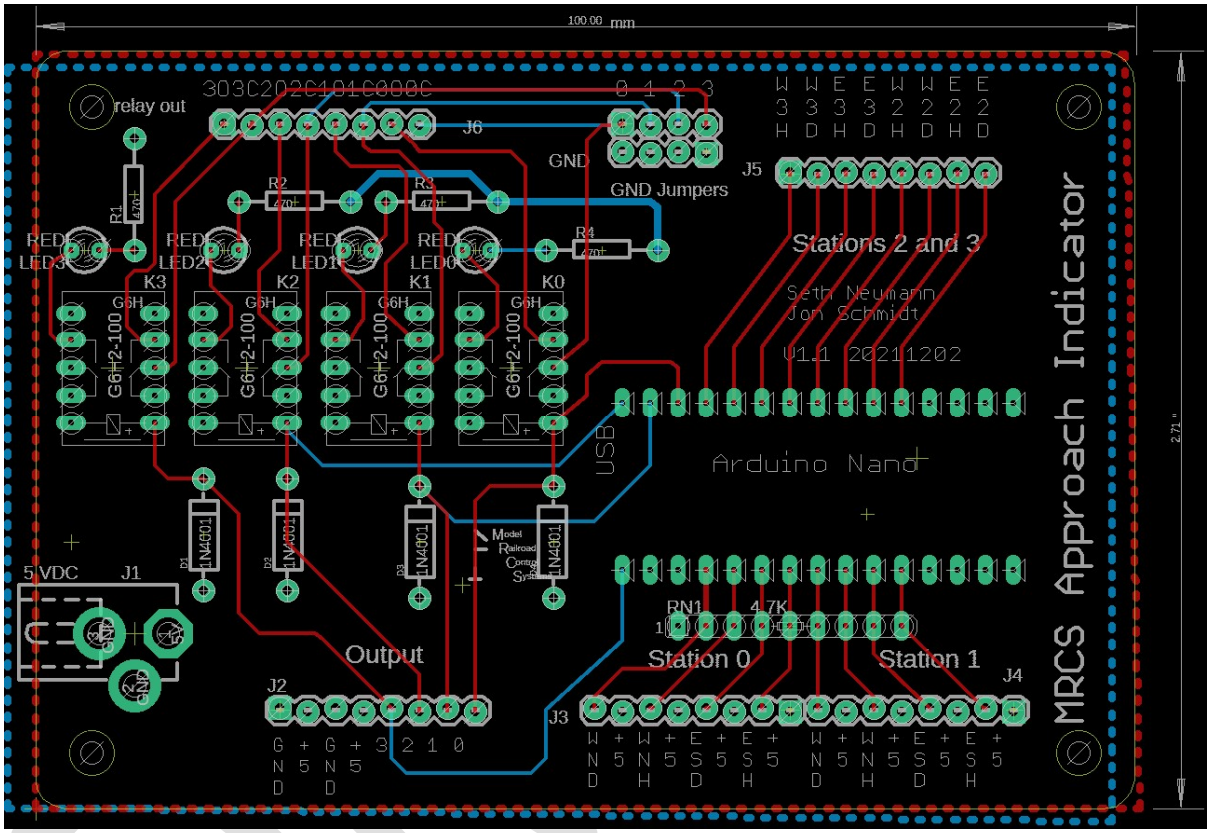


Figure 1 - Version 1.1 Board Layout

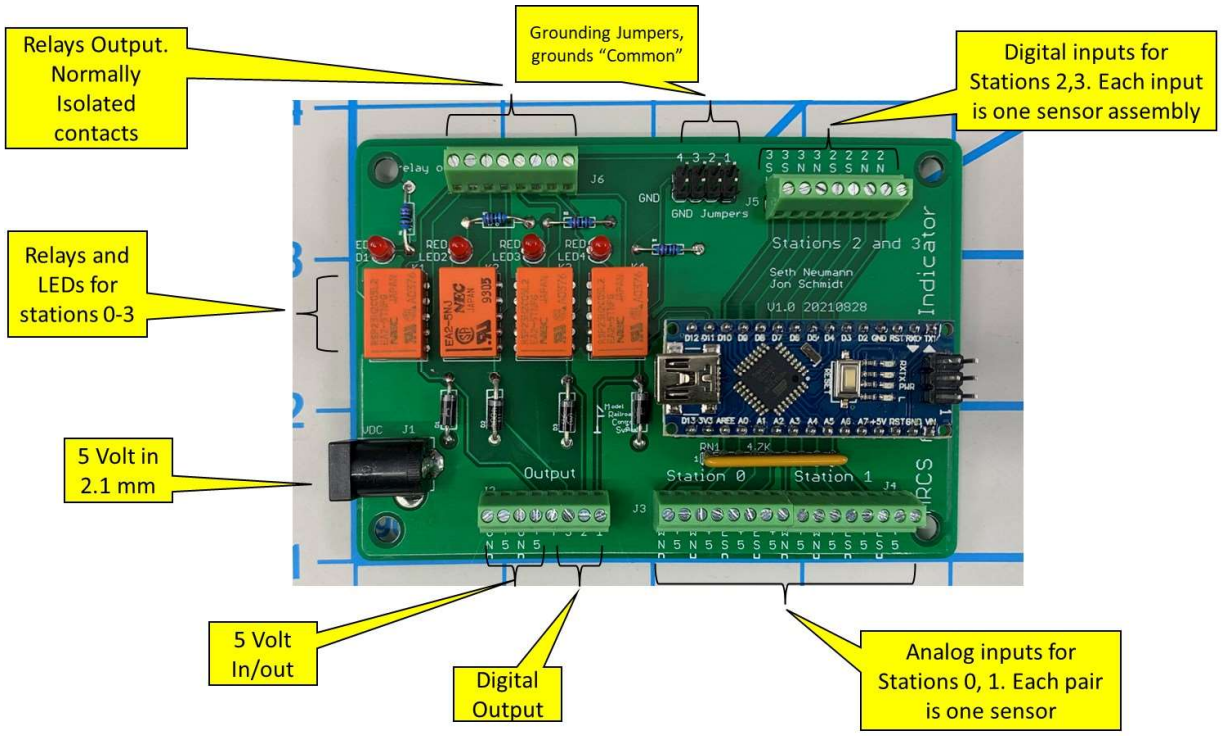


Figure 2 – Approach Indicator

This board is sold as a single unit, assembled, and tested or as bare board. If you are interested in alternate connectors, large quantities, or custom modifications please contact us. V1.0 shown.

2.2. BILL OF MATERIALS

Qty	Value	Device	Package	Parts	Description
4	1N4001	DIODEPTH	DIODE-1N4001	D1, D2, D3, D4	Diode
1	4.7K	G08R	SIL9	RN1	SIL RESISTOR
4	470	RPTH04	AXIAL-0.4-RES	R1, R2, R3, R4	Resistor
1	5 VDC	CONNECTOR-DC-POWER-RA	DCJ0202	J1	DC POWER JACK
4	G6H2-100	G6H2-100	G6H2-100	K1, K2, K3, K4	RELAY
1	GND	CONNECTOR-M04LOCK	1X04_LOCK	CON1	Header 4
1	GND Jumpers	CONNECTOR-M04LOCK	1X04_LOCK	CON2	Header 4
1	NANO	NANO	NANO	NANO	Arduino Nano
1	Output	CONNECTOR-M08LOCK	1X08_LOCK	J2	Header 8
4	RED	LEDLED3MM	LED3MM	LED1, LED2, LED3, LED4	
1	Station 0	CONNECTOR-M08LOCK	1X08_LOCK	J3	Header 8
1	Station 1	CONNECTOR-M08LOCK	1X08_LOCK	J4	Header 8
1	Stations 2 and 3	CONNECTOR-M08LOCK	1X08_LOCK	J5	Header 8
1	relay out	CONNECTOR-M08LOCK	1X08_LOCK	J6	Header 8
2	15 position	0.100 female connector	1x15	Arduino Sockets	Arduino Sockets
1	7x10	Circuit Board			Approach Indicator

Table 1- Bill of Materials Ver 1.0

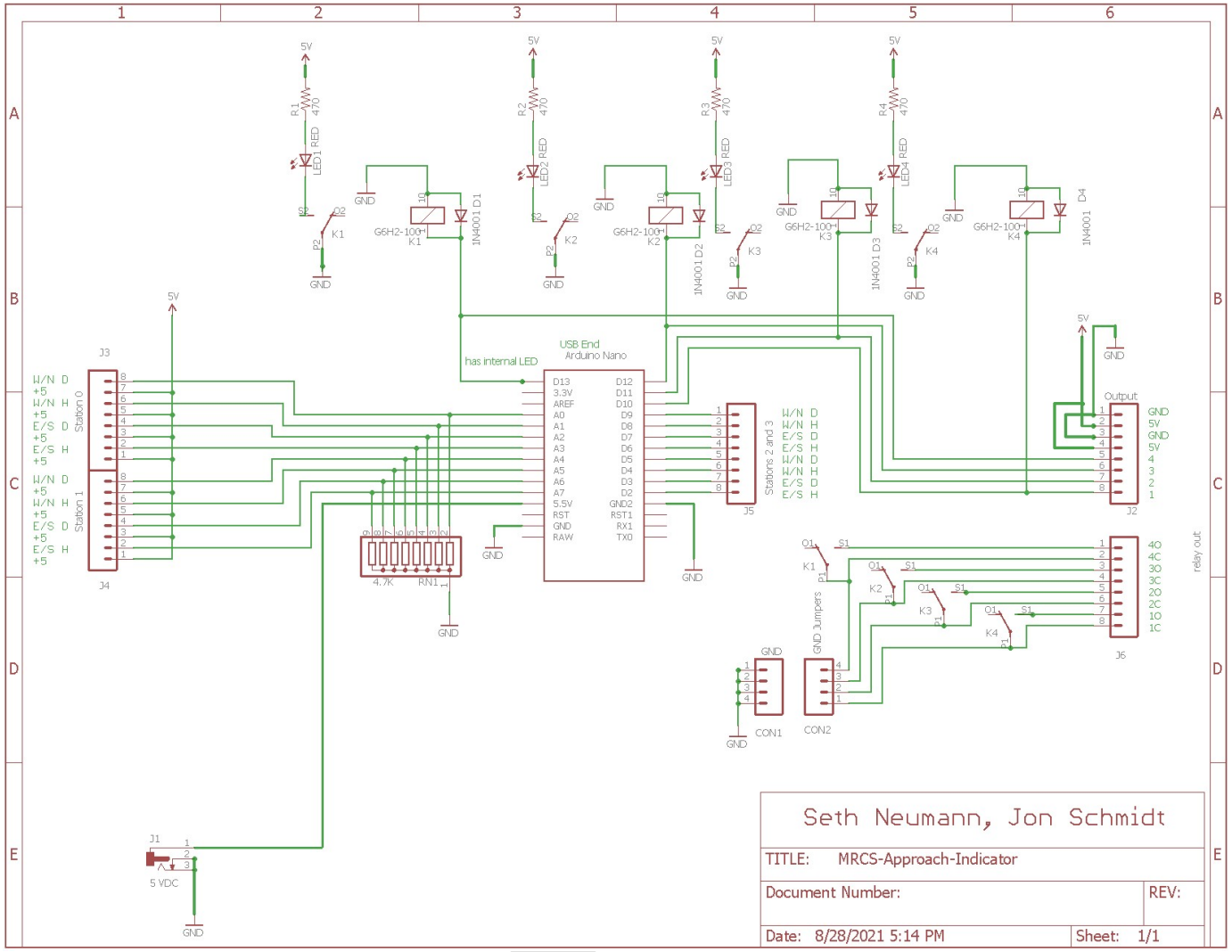


Figure 3 -Version 1.0 Schematic

3 OPTIONS

3.1. CONNECTORS

- The 8-position connectors are on 0.100 centers (staggered slightly to hold the connectors in place during assembly). While our standard connector is the 0.100 screw terminal, you may substitute any 0.100 connector you prefer. If you are ordering an assembled and tested unit from MRCS and you would prefer a different connector, please contact us at sales@modelrailroadcontrolsystems.com and indicate your preference and we'll provide a quotation.
- The Approach Indicator board has 7 sets of connectors, but not all of them are used in every application.

Connector	Type	
J1	2.1 mm DC power	Use this if you are using wall warts for power, if you are using the screw terminals on J2 you don't need this. However the 0.100 connections can be used to distribute 5V from the wall wart to other devices in the system.
J2	8-position 0.100 screw terminal	2 sets of +5/GND, + digital outputs for each indicator. These can be stuffed or not or use a 4-position terminal if you only need one or the other.
J3/J4	8-position 0.100 screw terminals	Inputs for optical sensors for stations 0 and 1, you only need to stuff the ones you need.
J5	8-position 0.100 screw terminal	Digital Inputs for Stations 2, 3.
J6	8-position 0.100 screw terminal	Outputs from Relays, isolated contact per. If you're only using the analog inputs, you can use a 4 position terminal
GND Jumpers	2x0.100 male header	Use a berg jumper to ground the Common side, so the commons become grounds and the "opens" are grounded when the relay is active

Standard configurations (offered on our website without special order) are bare board, 2 station analog only, and fully implemented. Please contact us if you want a different configuration assembled and tested.

3.2. LEDS

- We use Red LEDs with 470 ohm limiting resistors for the on board LEDs (used as indicators of relay status), there is no magic to this: any color will work fine. The supply is 5V. In case you are new to building up electronics, here's a handy table of typical LED colors and values, the more experienced reader will have his/her own preferences and whatever you have on the bench is probably fine!

- If you want remote LEDs, you can mount a 2 position 0.100 header in the LEDs connections and use that to connect to a remote LED, the on-board limiting resistor will protect it.

Color	Forward drop	Typ Current	Suggested Resistor value*
Red	1.5V	10mA	330 ohms
Yellow	1.8V	10mA	330 ohms
Green	1.8V	10mA	330 ohms
White	3.0V	0.5 mA	4.7K ohms
Blue	3.0V	0.5 mA	4.7K ohms

The formula is supply voltage (5) – Forward Drop (1.5v for Red) = 3.5/current in A, = .01 = 350 ohms. * Using nearest standard value.

Table 2 - Suggested LED limiting resistor values

3.3. RELAYS

The board is designed for G6H2 relays as the 5 volt variant can be driven directly by an Arduino output. If you are building the approach indicator up from a circuit board and want to use some other style of relay, you should connect the digital outputs, which are brought to the J2 terminal block and connect them to your favorite relay driver circuit. "Arduino" relays may be connected directly to the digital outputs.

Each relay circuit consists of a relay, a "snubber" diode to protect the Arduino pin driving it, an LED with current limiting resistor to protect it. The relays are connected to the stations as follows:

Relay	**Station**
K1	0 (Analog)
K2	1 (Analog)
K3	2 (Digital)
K4	3 (Digital)

Table 3 – relays

There is no need to install relays for stations you don't intend to use. The LEDs are handy for checking operation but can be omitted if not needed.

4 ASSEMBLY

4.1. DETAILED ASSEMBLY INSTRUCTIONS

See "Options" above and refer to table 1 "Bill of Materials" and determine which connectors, relays and other components you will use and mark this list to avoid confusion during assembly. Only install the parts you need.

[ ] All of the components are through-hole technology with wire leads. A lead bender is a useful tool for forming the leads at 90 degrees for easy insertion into the pad holes. See "suggested tools" below if you are new to assembly.

The general rule is to install the lowest components first, working towards components that are higher off the board. This enables you to support the low components as you solder them. Most of the components will stay in place as you flip the board over to solder but if not use a small piece of cardboard to hold them in place as you flip the board over. Use 0.015 solder unless otherwise indicated to help control the flow of solder onto the work.

[ ] Resistors

[ ] Install R1,2,3, 4. Use your lead bender to make 0.400 bends. There is no polarity but for consistency I put the gold "tolerance" band towards the Arduino or towards the bottom of the board.

[ ] Snubber Diodes

[ ] install diodes D1, D2, D3, D4. with the cathodes (band) up.

[ ] Resistor pack

[ ] install the resistor pack RP1 (pull downs for the sensors) with the common (dot) to pin 1 on the left.

[ ] LEDs

[ ] install 3mm LEDs 1, 2, 3, 4 align the flats on the Left side. See section 3.2 above for typical values. (If you're using remote LEDs, wait to install the headers until after the relays).

[ ] Relays

[ ] install relays K1, K2, K3, K4 with the bar on the package down.

[ ] 15 position female headers for Arduino Nano

[ ] install the female headers, use a Nano (they usually come with the male headers installed) to hold the female headers in alignment. A properly grounded electronic soldering iron (see tools below) will not harm the Nano, but experienced assemblers will have a dead Nano around to use as a fixture.

[ ] Screw Terminal

[ ] Install the 8 position screw terminals J2, J3, J4, J5 and J6 with wire openings pointing off the board.

[ ] Install Ground Jumper Headers

[ ] install the 2 x 4 pin male headers, you can use 2 1x4s or 1 2x4,

[ ] Install the DC Power Jack J1

[ ] Use 0.031 solder for this. Fill the holes with solder so there are no gaps. You can mount the servos in either orientation.

THIS WOULD BE A GOOD TIME TO MAKE SURE YOU DON'T HAVE ANY SHORTS!

Use your bench supply with a banana jack to 2.1 mm power adapter and apply 5V. (a 5V wall wart with your volt-ammeter in series will do also, a few clip leads will be handy.) You should not be drawing any current and the output of the supply should hold at 5V. If it's drawing current, look for solder bridges or pins that haven't been soldered.

[ ] install the Nano

Insert the Nano into the two female headers. (I'm assuming you've already programmed it) with the USB connector facing towards the DC power Jack (the board says "USB"). Make sure the headers are lined up (not spilling over one end). Try powering up again, you should be drawing about 20 mA with no relays on. Expect about an additional 30mA per active relay.

4.2. LIST OF HANDY TOOLS FOR THE CIRCUIT BOARD ASSEMBLER

- Lead Bender
- Side Cutters
- Small Needle nose Pliers
- Temperature controlled solder iron. Don't skimp here, this is a very useful tool for everything you do in model railroad electronics, get one with replaceable tips, the finer the better Weller (such as WE1010 NA) and Hako (FX888D) make very nice irons that balance well in your hand for <$150 both available from Digikey and Amazon.
- 0.015 solder for fine pitch items like 0.100 connectors (also handy on decoders)
- 0.031 solder for larger items like the power connector
- Isopropyl alcohol 91% or 99% for cleaning left over flux off the board
- Bench power supply – this gives you precise control of the voltage and allows you to measure and limit current as you test. You can see if you are drawing the correct amount of current, if not that's an indication that something is wrong. I like these: https://www.banggood.com/Topshak-NPS3010W-110V-or-220V-Digital-Adjustable-DC-Power-Supply-0-30V-0-10A-300W-Regulated-Laboratory-Switching-Power-Supply-p-1474957.html?cur_warehouse=CN&rmmds=search Equivalent units are available from many suppliers.
- Digital Volt-Ohm-Milliamp meter. This really comes down to features like quality of the probes and how the stand works, Harbor Freight. Marlon Jones and many off-shore suppliers have very functional meters for as little as $10.

5 SOFTWARE AND PROGRAMMING THE ARDUINO NANO

The Arduino Nano is programmed using Jon Schmidt's "Approach Indicator 5.1" sketch using the free Arduino Integrated Development Environment (IDE). You can get it at https://www.arduino.cc/en/software. The IDE is available for Windows, Mac and Linux. There are many tutorials and guides on how to download and install the Arduino IDE if it doesn't just work for you. Note that most commodity Nanos use the CH340 serial chip and you may need to find and download the driver. They usually are supplied by the Asian vendors who make the chips and look pretty bare bones, but they work!

When you start the IDE, go to the "tools" menu:

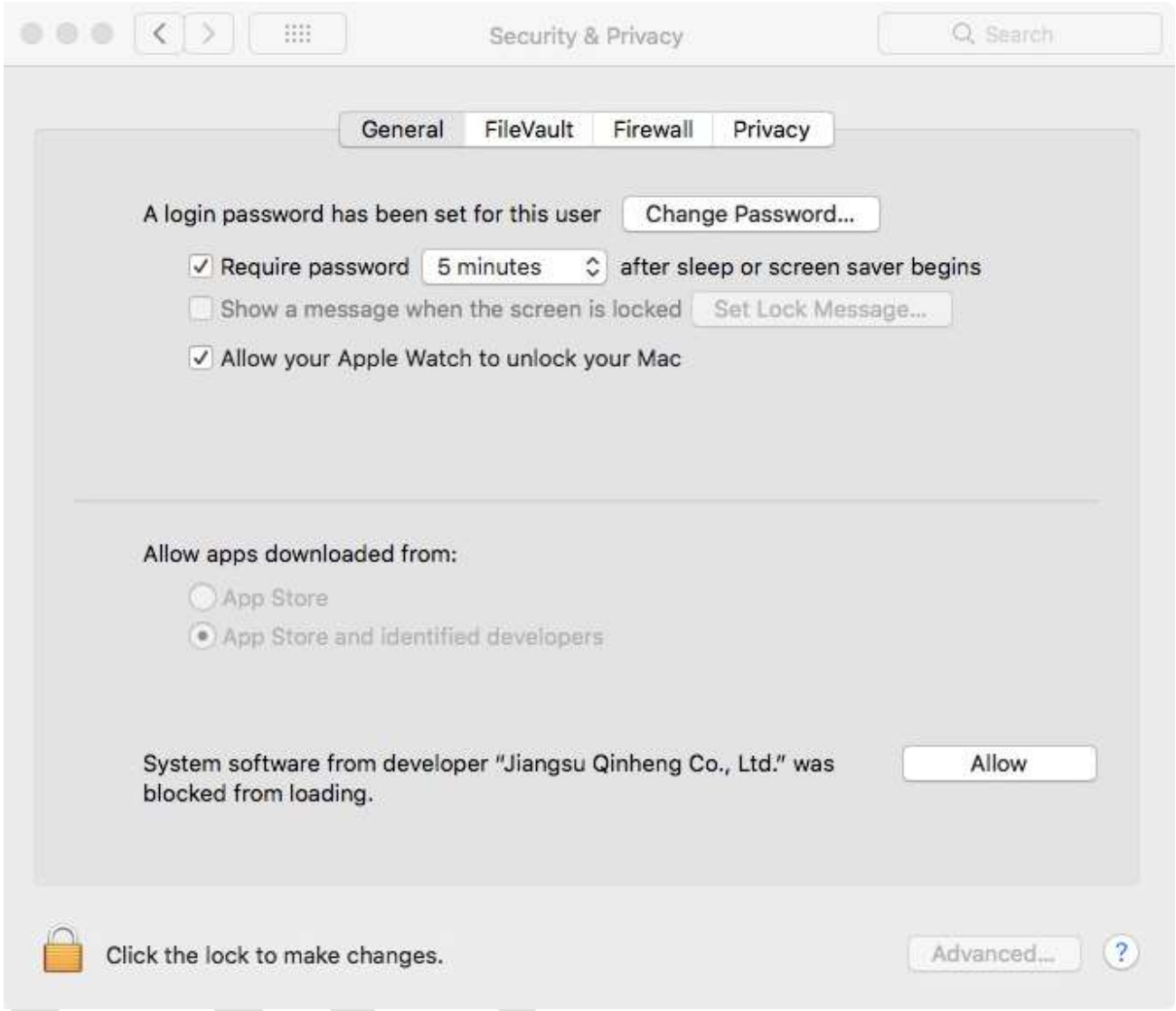
I've use the following settings on my Windows 10 machines.

Go to Board, mouse right and select "Arduino AVR Boards" and select "Arduino Nano"
Go to Processor, mouse over to the right and select "ATmega328P (old bootloader)"
Go to Programmer, mouse over to the right and select "AVRISP mkii"
Go to Port and be sure it's not COM1 (if that's the only choice you'll have to install the CH340 driver, or at least reboot to make your machine see it).

A Mac friend suggests these for the Mac:

Download the Arduino IDE from https://www.arduino.cc/en/Main/Donate
double click the arduino-1.8.5-macosx.zip file and drag Arduino.app into your Applications folder
Download serial driver from https://github.com/adrianmihalko/ch340g-ch34g-ch34x-mac-os-x-driver
double click the ch340g-ch34g-ch34x-mac-os-x-driver-master.zip file
double click the CH34x_Install_V1.4.pkg file to begin installation

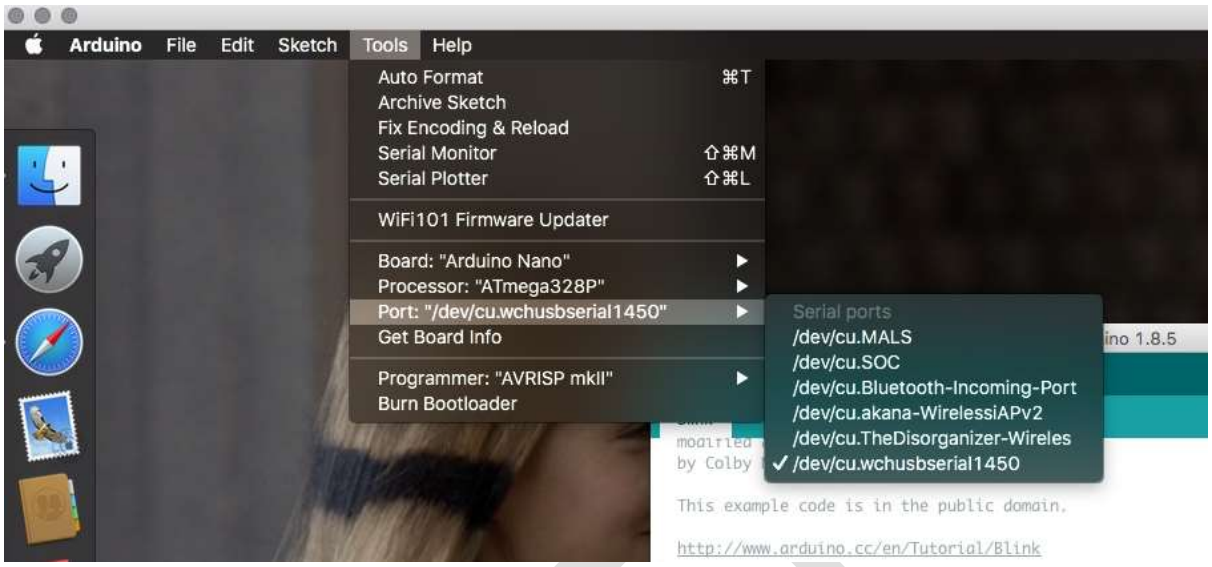
click Open Security Preferences



Click Allow
Restart your Mac - it takes a little time to rebuild the driver cache - don't panic!

Run Arduino.app from your Applications folder. There will be a prompt whether you "really want to do this". Say ok - because hey - we've got a grade crossing to program!

in the Tools Menu set the port to /dev/cu.wchuserial1450 (see below)



To prove to myself this worked I followed the instructions at https://www.arduino.cc/en/Guide/ArduinoNano to try loading the "Blink" sketch under the heading "Open your first sketch". Once you upload it the board red light will blink regularly.

If you're using Linux, you can probably figure this out for yourself.

Since the D0, D1 serial data lines on the Nano are not connected to anything, it's OK to leave the Nano on the Approach Indicator while reprogramming.

6 USING AND MODIFYING THE APPROACH INDICATOR SKETCH

6.1. DESIGN:

The Approach Indicator board is designed to create an alert based on the approach of a train. It relies on pairs of sensors in the track or trackside to detect the presence and direction of the oncoming train. Each "station" has two "approaches" or pairs of sensors, one pair for an east arrival and one pair for a west arrival (or north/south or left/right).

Features:

- Up to four stations (eight approaches) per board.
  - Two stations with analog sensor sets.
  - Two stations with digital sensor sets.
- Up to four alert outputs per board, with digital signal output or relay contacts.
- Analog sensor sets are self-calibrating on a timed cycle to adjust to changes in room lighting (as the sun moves, for example).

Possible configurations per board:

- Single station, single alert output.
- Single station, two different alerts signaling arrival direction.
- One to four stations, each with its own alert output.
- One to four stations, common alert output. Or a mix.

The sensors should be installed in the center of the track bed approximately where you want the alert to occur. Each pair should be no further apart than the smallest object you want to detect. If you want to be able to detect the arrival of a 5-inch locomotive, the sensors must be no farther apart than 5 inches. We recommend 6 to 10 inches for most situations.

6.1.1. Software operation:

The sketch has many user-configurable definitions. These are described in a later section. The operation of the sketch depends on many timers most of which are easily configured.

The main loop proceeds as follows:

For each configured station:

Check if any older alerts need to be cleared and clear them.

Read the analog sensors and integrate over a specified time to average the effect of 50 or 60 hertz power affecting light output.

Read the digital sensors. Invert the signal meaning if necessary.

For each direction check if the direction for that station is eligible for an alert, based on the time of the last alert generated for that direction for that station.

If the direction is eligible check the sensors: Both must have been triggered, and the home sensor must have been triggered on or after the time that the distant sensor was triggered.

If so, trigger the alert and set timers.

Check if it is time to recalibrate the analog sensors and flag them to recalibrate.

6.2. APPROACH INDICATOR SETUP

Here are the basic definitions within the sketch that may need modification:

```
// Define the active stations: 0-1 Analog detection, 2-3 digital detection
// A "station" is a set of two approaches, four sensors:
//                    East DISTANT/HOME and West DISTANT/HOME
//                    Station 0   Station 1     Station 2   Station 3
//                    East  West  East  West    East   West   East   West
const boolean WorkApproach [8] =
                    {true, true, true, true,  false, false, false, false};//
```

WorkApproach controls which sets of sensors will be interrogated. We recommend that sets which are set to true must have sensors connected. Open digital sensors may present false positives.

```
// digital output pins for alert relays for each approach
//                   Station 0   Station 1     Station 2   Station 3
//                   East  West  East  West    East   West   East   West
const int AlrtPins[8] =
                   {10,   10,   11,   11,    12,    12,    13,    13};
```

AlrtPins controls which output pins are activated on an alert. Each approach can have its own output, or sets can be combined to a single output.

```
// duration of alert in seconds - how long to keep signal high
const int AlertLenSecs[8] =
                   {10,  10,  10,  10,  10,  10,  10,  10};

// dont repeat alert for a bit - in seconds - stop false repeats
#define DontRepeatSecs 30
```

DontRepeatSecs blocks generating a new alert for an approach. It does not block an alert for that station coming from a different direction.

```
// hold occupied status for this time to minimize bounce (false clear)
// - for couplers, skeleton cars - in milliseconds
#define HoldOccMs 1000
```

HoldOccMs keeps a detector signal *true* in the software for the specified time after the last time the signal was physically *true*.

```
// if SnsrActiveLED is non-zero, trigger the output indicated if any sensor is active
// -- useful during setup – note possible conflict with AlrtPins
//#define SnsrActiveLED 13
#define SnsrActiveLED 0
```

SnsrActiveLED is most useful during installation. Associating it with an output signal provides a physical manifestation that a sensor has been covered. Use it for confirming connections. A later discussion of the *Trace* facility shows how to identify specific pin activation.

```
// delay in ms for main loop sampling
// -- 0 = continuous
// -- large value if tracing or debugging
#define MainLoopDlyMS 0
```

MainLoopDlyMS should be zero for live operation. Set it to a large value (5000 – 10000: 5 to 10 seconds) only if Trace is enabled and you want to read the generated output on the IDE Serial Monitor.

6.2.1. Analog Sensors

```
// Analog sensor pins
// NOTE a sensor entry of 0 in the *pins arrays will be skipped
// D—distant  H--home
//                    Station 0        Station 1
//                    East    West     East    West
//                    D  H    D  H     D  H    D  H
const int Apins[8] = {A0,A1,  A2,A3,   A4,A5,  A6,A7};// analog sensors - Nano
```

Apins lists the analog sensor pins. Note that the order of the pins for each set must remain ND-distant, NN-near, SD-distant, SN-near (north/south). If there is no connection to a pin, either disable the entire set via ***UseSet*** or set the entry in ***Apins*** to 0.

```
// analog sampling controls - needed to average light flicker
// length of sample integration in millisecs
// 34 ms for 60 cycle lighting mains
// 40 ms for 50 cycle lighting mains
#define LoopMS 34
```

LoopMS specifies how long the analog data collection will take for each cycle. The sensors for the set are continuously interrogated over the time frame, and the results are averaged. This will fix "flicker" of quick-response light sources such as LEDs.

```
// delay within integration in millisecs 0 = no delay
#define DlyInLoop 0
```

DlyInLoop adds a delay to the analog sensor data collection. We recommend 0.

```
// analog sensors are recalibrated occasionally to account for changes in lighting
//  such as sunlight moving through a window
// minutes between recalibrate low/high analog sensor values
#define RecalibrateMts 5
```

RecalibrateMts forces the analog sensors to recalibrate every specified period.

6.2.2. Digital Sensors

```
// Digital sensor pins
//                    Station 2   Station 3
//                    East  West  East  West
//                    D  H  D  H  D  H  D  H
const int Dpins[8] = {2, 3, 4, 5, 6, 7, 8, 9};// digital sensors
```

Dpins lists the digital sensor pins. If there is no connection to a pin, either disable the entire set via ***WorkApproach*** or set the entry in ***Dpins*** to 0 otherwise a false positive may result.

```
// invert flag for digital sensor if HIGH sensor means clear
boolean   InvertDig = true;
```

InvertDig sets the software to invert the meaning of the digital signal. *HIGH* is normally *true*. ***InvertDig*** reverses that meaning if the hardware requires it.

6.2.3. Trace/Debug

```
// ****************** Trace & Debug
// 0 - no trace; 1 - flow; 3 - everything; 4 - too much
#define Trace 0
```

Trace enables output to the IDE Serial Monitor. Please refer to the sketch itself for details.

6.3. HELP INSTALL AI SETUP

There is an additional sketch which may be used to assist in installation of Approach Indicator. The sketch will drive the board to produce an output signal when a sensor is triggered. This way one can confirm that a sensor is a) working, and b) connected to the proper input.

The HelpInstallAI sketch will:

- First, trigger each defined output.
- Loop continuously, checking each defined set and its associated sensors.
- If a sensor is triggered, the sketch will trigger the associated output a number of times:
  - Report the set number +1
  - Report the line number +1

Here are the basic definitions within the sketch that may need modification:

```
// NOTE: The program may be configured to monitor 4 sensor sets/towers/bells
//       This would require 2 sets of analog and 2 sets of digital sensors
boolean UseSet [4] = {true,true,false,false};
//
// digital output pins for alerts for each set
```

```
const int Alrts[4] = {10,11,12,13};
//
// duration of alerts in milliseconds
#define SetAlertMs  1000
#define LineAlertMs  400
//
// Analog sensor pins
// NOTE a sensor entry of 0 in the *pins arrays will be skipped
//             Set0   ND NN SD SN  Set1 ND NN SD SN
const int Apins[8] = {A0,A1,A2,A3,      A4,A5,A6,A7};// analog sensors - Nano
//
// Digital sensor pins
//             Set2   ND NN SD SN  Set3 ND NN SD SN
const int Dpins[8] = { 2, 3, 4, 5,       6, 7, 8, 9};// digital sensors
// invert flag for digital sensor if HIGH sensor means clear
boolean   InvertDig = true;
//
// analog sampling controls - needed to average light flicker
// length of sample integration in millisecs
// 34 ms for 60 cycle lighting mains
// 40 ms for 50 cycle lighting mains
#define LoopMS 34
//
// analog sensors are recalibrated frequently to account for changes in lighting
//  such as sunlight moving
// minutes between recalibrate low/high analog sensor values
#define RecalibrateMts 5
//
// delay in ms for main loop sampling
// -- 0 = continuous
#define MainLoopDlyMS 5000
//
// echo to serial monitor
#define Trace 1
```

7 TESTING

Testing your Approach Indicator is straight forward:

1. Connect 4 (PT-204) sensors (Digikey part 1080-1019=ND) to station 0. The connections are West/North Distant – WND – and +5, West/North Home and +5, East/South Home and +5, and East/South Distant. Note the "flag" side (short lead) goes to +5 (opposite from an LED).
2. Block the light on a Distant sensor, and then also block the light on the Home sensor. Relay K1 should operate and its LED will light. It should time out after about 4 Seconds. Repeat for the other direction. If you want to change the timeout, search for "#define AlertLenMs 4000" and change the time value (in milliseconds).
3. Repeat for station 1.
4. For the digital stations (2,3), apply a ground to the Distant and then Home inputs, simulating a detection signal from the external detector assembly.

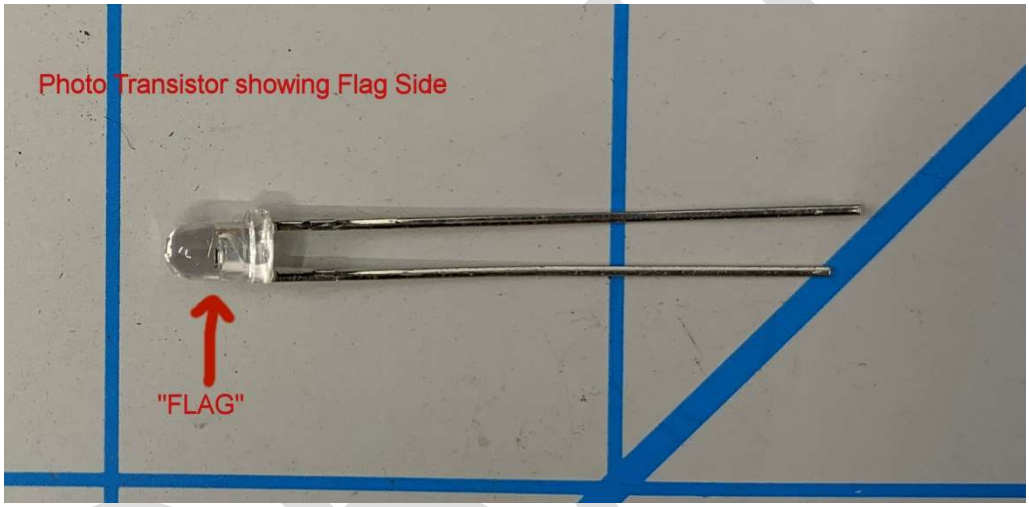


Figure 4 - Photo Transistor

8 INSTALLATION AND CONNECTIONS

8.1. POWER CONNECTIONS

Power is 5 Volts DC. The board has a 2.1mm barrel jack for DC power (typically from a wall wart, but you could tap off a local 5V bus or use our "Buckeroo" power distribution board to derive 5V from a 12 V bus. You can also supply 5V to the 5V and ground terminals of the 8 position screw terminal block on J2. If you have multiple Approach Indicator boards in the same area, you can daisy chain from one to another assuming the power supply can source enough current.

8.2. SENSOR CONNECTIONS

8.2.1. Analog Sensors (stations 0,1)

Connect 4 (PT-204) sensors (Digikey part 1080-1019-ND) to station 0. The connections are West/North Distant – WND – and +5, West/North Home and +5, East/South Home and +5, and East/South Distant. Note the "flag" side (short lead) goes to +5 (opposite from an LED). See notes under "Testing" above.

8.2.2. Digital Connections

Connect the digital outputs of sensors to the inputs on J5, home and distant in each direction. These sensors may be optical units such as out Enhanced Optical Position Detector (EOPD), our forthcoming Simple Optical Detector or units from many Model RR oriented suppliers or commercial units from off-shore sources. Be sure the grounds of the Approach Indicator and the detection units are connected. If the detector uses *LOW* as a *true* (occupied) indication, search for and change the `InvertDig` definition to *true*.

8.3. OUTPUT CONNECTIONS

Output connections are made on J6 for relay outputs, or J2 for signal outputs. Each relay has an isolated, normally open contact (labelled n0, c0 where n is the normally open contact and c is common). These can be used to route power or another signal (such as audio) to whatever is connected to the other side. Since many applications expect to see a ground when the relay operates, the jumper block allows you to connect the common to the board's ground, the n(n) contact will be grounded when the relay is operated.

Here's a buzzer we like, connect the black wire to n(n) and the red wire to 12V supply (the 6 volt version of the buzzer is pretty wimpy).

The default "bell time" in the sketch is 4000 mS (4 seconds), it can be changed in the sketch by searching for "#define AlertLenMs 4000" and editing the time value (in milliseconds).

Figure 5 – Vellman SvM12N buzzer (12VDC)

http://www.modelrailroadcontrolsystems.com/dc-buzzer/

Vellman SvM12N buzzer, Red to +12V, Black to ground, draws ~30mA. Available from your favorite electronic supplier.

Another common application is to trigger a sound module with a recording of your favorite bell. We offer an the excellent CTC Bell SoundBytes module from Iowa Scaled Engineering

Figure 6 - Iowa Scaled SoundByte

CTC Bell SoundBytes module

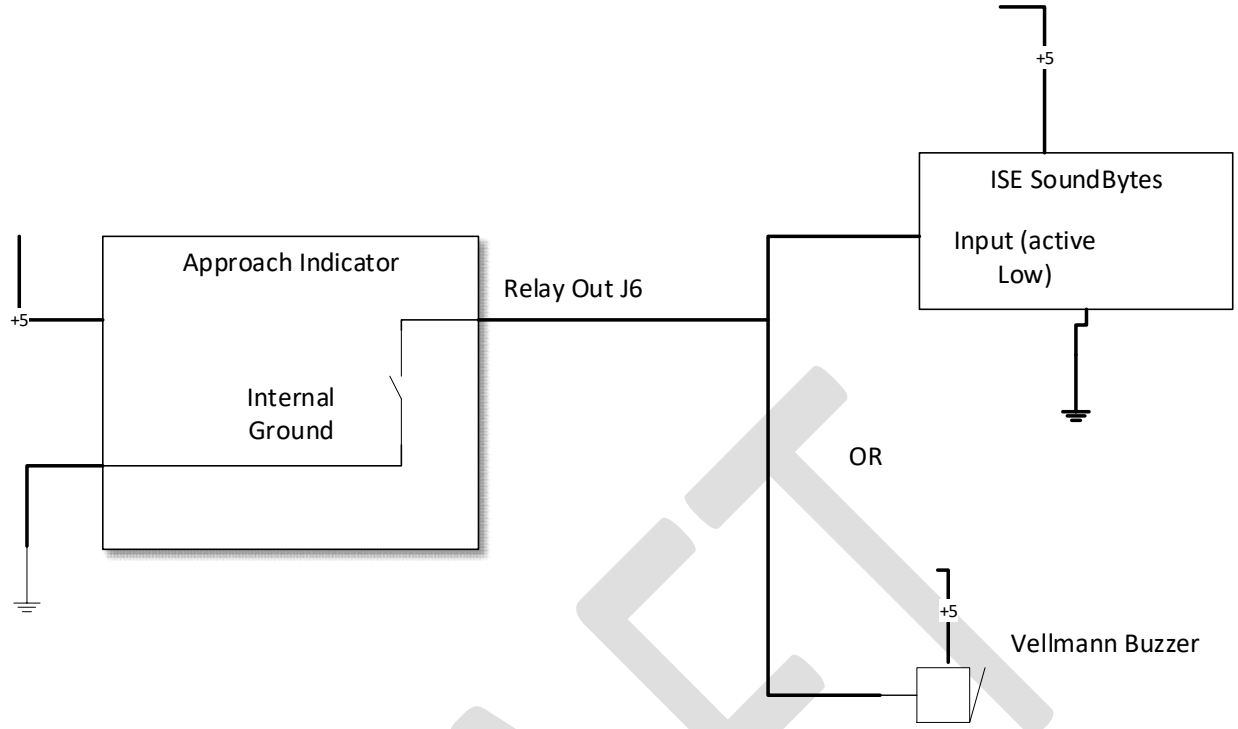


Figure 7 - Output Connections to Buzzer/Sound Module