

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

Hi All,

I take great pleasure in introducing **Mike Jagoe's** outline article to build an Arduino 8 port s88 to solve a problem on his layout.

Mike first contacted me in February 2018 re some problems he was having with his turntable and we have remained in contact since then.

Mike told me about his Arduino project and I asked him if he would like to write it up and post it on my Web site. My thanks to Mike for contributing to my Tips page.



Mike's Article Below

Arduino based S88 stop markers for ECoS/Train Controller environment

My layout is Märklin c track with an ECoS controller using ESU detectors (both current and ground) for detection, ESU Switch Pilots and Servo Pilots. Some two years ago, I added a two-rail trolley line to the front of my layout to run two San Francisco PCC trolley cars in shuttle mode. The two trolley cars share the track, run in opposite direction to each other and pass with very tight quarters in the centre of my layout. This shuttle line is a two-rail setup with current detection.

The problem is getting consistent stopping at the passing point. On the two-rail track I use a single current sensed piece of track for each block and use TC to calculate the required stopping distance. When I did the block occupancy on the C track, Ross strongly advised me to use 3 sensors per block (for bi-directional running), which I did. In the course of working with the C track layout and Train Controller I came to appreciate that (for TC) there is a huge difference between a single sensor (for ramping and stopping) and a setup with two sensors for the ramp point and the stop point. With a single sensor, any sort of track dirt causing a 'hiccup' in the loco travel will cause an error (as time passes and the loco is not moving). In fact, with the trolley, it was possible for it to stall with its tail still on the switch – an unacceptable outcome.

My detectors are fully utilized (no free ports). So, I was scouting around for a solution to get an additional stop sensor into my trolley track and hence a signal to the ECoS that the stopping point has been reached. I had always kept in mind that I had seen Rudy Boor's Arduino based S88 project and I had a couple of Arduino UNO's laying around, so I set out to determine if I could make a system to give me a stop signal.

Disclaimer: I am not an Electrical or Electronics engineer or technician! You use any of the following at your own risk. I am a simple hacker who enjoys the challenge. Under no circumstances should you try to use this project to connect directly to your track ground for a S88 Märklin track occupancy detector – the Arduino will be destroyed.

Arduino information: A 'getting started guide' See links below

<https://www.arduino.cc/en/Guide/HomePage>

I use the Arduino on line IDE coding environment

<https://www.arduino.cc/en/main/create>

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

Best approach is to get an Arduino, follow the guide and install a program to make the on-board LED blink – Once you are able to accomplish that, you should be able to move onto this level of project. Just remember it is a ‘one way’ push of code to the Arduino – you can never read what code is installed on an Arduino, simply load new code and start again.

To get my code into the editor, simply open the supplied file **RB_S88_LDR.ino** with the Arduino on line editor or copy from **Appendix A** on pages 11-13 and then paste into the Arduino on line editor.

Source Information: Rudy’s site is a great resource for Train Controller and Arduino based projects.

<https://rudysmodelrailway.wordpress.com/>

Two links to his S88 videos:

[First test with sensors on the track and Arduino s88](#)

[Multiple Arduino’s chained as an s88 occupancy detector interface](#)

Rudy’s software is available here:

[RB-S88-v03.zip](#)

Rudy’s S88 project is based on a two-rail layout and he is using reed switches set in the rails and magnets on his locos to trip the reed switches. I had prior experience using LDR’s from one of my very first Arduino projects which was to build a speed trap for my layout by following this project exactly (no changes were required).

[An Arduino Powered \(Scale\) Speed Trap](#)

I found this worked very well (and reliably) and I had a bunch of LDR’s left over. I had also built a derivative circuit of this for my grandsons Thomas train layout. In this case I built a state machine with 4 signal masts, an LDR in the wooden track at each mast. As each mast is passed, the LDR is switched and the software turns the current signal yellow, the next one in front turns green, the remaining two turn red.

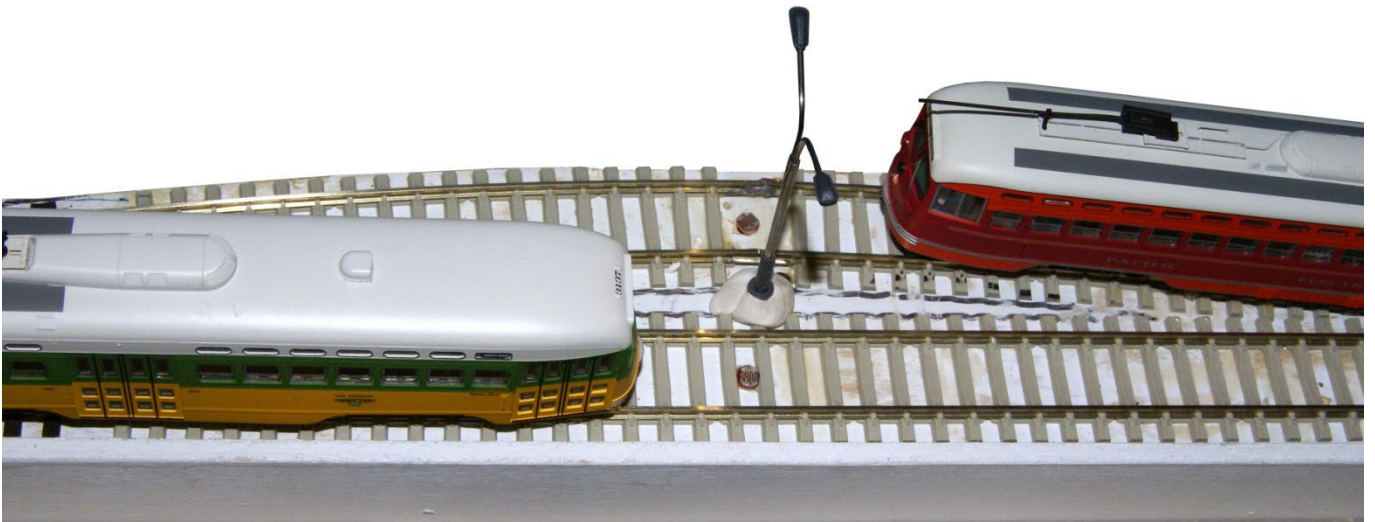
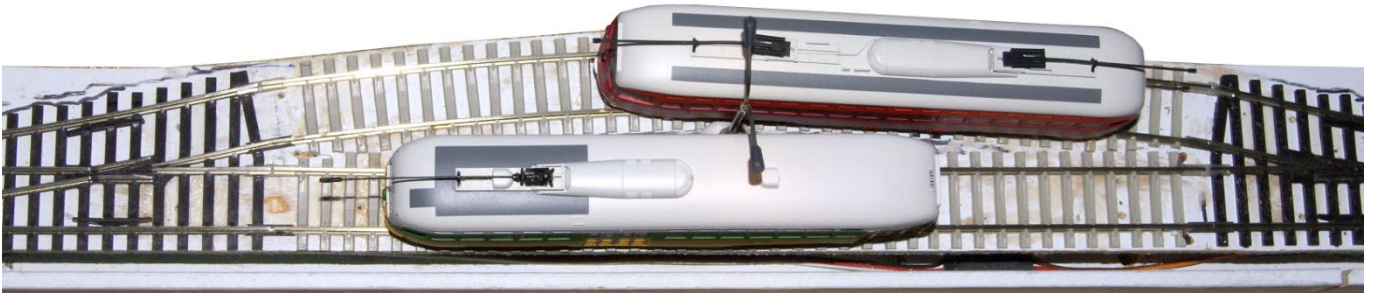
So, based on all this I embarked on my project:

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

My Project

1. Here are some photos of the close quarters that the trolleys have to negotiate to pass:



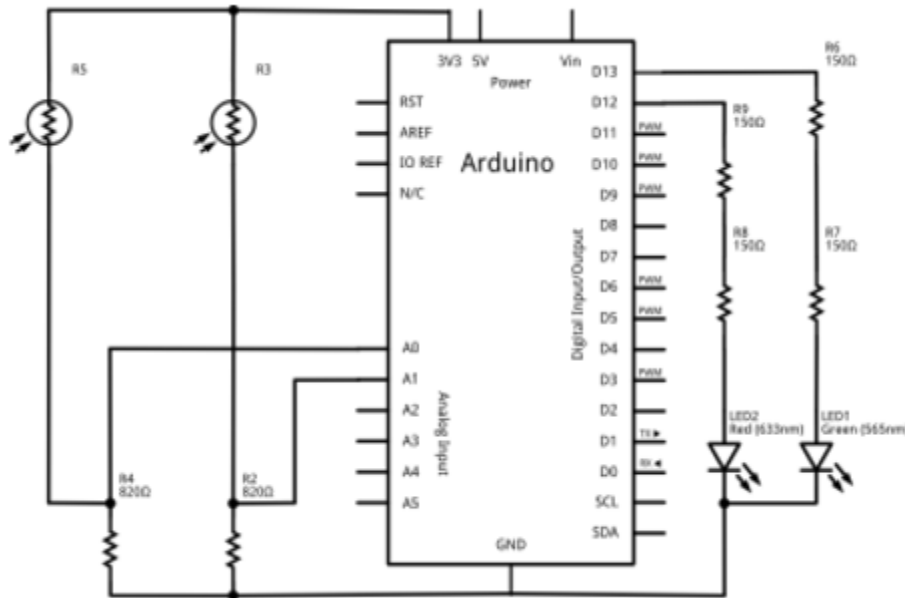
2. Sort out my LDR's and select 4 similar devices 3.6k ohms ambient light; 30k ohms covered.
3. Construct a test setup on the layout and install a lamp standard that will provide a constant light source over the LDR's to avoid errant switching.



Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

- Here we can see the two LDR's installed and the 12v two led flood light installed. I am happy with the level of light on the LDR's.
- Test my simplified speed trap/Thomas train LDR code to ensure I get the LDR's calibrated and switching as desired on the trolley line. The breadboard circuit is built right off the speed trap circuit diagram but I will not be using the RHS connections for LEDs.



- The Arduino code is shown in the LHS of the table in the **Appendix B** pages 15-17 “Combine the LDR software with the S88 software” (snippet shown below) and the output on the screen monitor including a ‘finger cover’ test of tripping the LDR's is as follows:

<p>Newline ▾ 9600 baud ▾</p> <p>SEND</p> <pre>Calibrating Sensor Sensor 1 123 Sensor 2 82 Sensor 3 1024 Sensor 4 1024 DoneSensor 1 triggered60 0 Sensor 1 triggered59 0 Sensor 1 triggered56 0 Sensor 1 triggered48 0 Sensor 1 triggered44 0 Sensor 1 triggered41 0 Sensor 1 triggered41 0 Sensor 1 triggered40 0 Sensor 1 triggered46 0 Sensor 1 triggered50 0 Sensor 1 triggered52 0 Sensor 2 triggered40 1 Sensor 2 triggered39 1 Sensor 2 triggered37 1</pre>	<pre>Signals_test_ofLDR_Trol ▾ Serial.println("Calibrating Sensor"); // calibrating sensors: while (millis() < 5000) { //for the first five seconds check and store the lowest light //level seen on each sensor sensor1 = min(sensor1, analogRead(SENSOR_1)); sensor2 = min(sensor2, analogRead(SENSOR_2)); //sensor3 = min(sensor3, analogRead(SENSOR_3)); //sensor4 = min(sensor4, analogRead(SENSOR_4)); } Serial.print("Sensor 1 "); Serial.println(sensor1); Serial.print("Sensor 2 "); Serial.println(sensor2); Serial.print("Sensor 3 "); Serial.println(sensor3); Serial.print("Sensor 4 "); Serial.println(sensor4); delay (5000); //the cut off level for triggering the state machine //is half the resistance seen during calibration sensor1 = sensor1 * 0.5; sensor2 = sensor2 * 0.5; // sensor3 = sensor3 * 0.5; // sensor4 = sensor4 * 0.5; Serial.println("Done");</pre>
---	--

So, that looks promising – now I should test Rudy's code, so I know I have it working.

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

- Next issue is to build a cable to connect the pins on the Arduino to the ECoS. Here is a copy of what Rudy has defined, with my planned wire colors noted:

Connections for S88 bus:

BLU - s88 pin 1 Data - Arduino pin 13 = Data_Out to Command Station, or to the previous Arduino in the chain

GRY-BLK - s88 pin 2 GND - Arduino GND

BRN - s88 pin 3 Clock - Arduino pin 2, interrupt 0

YEL - s88 pin 4 PS - Arduino pin 3, interrupt 1

S66 pin 5 Reset (not used here) - Arduino pin 12, used as DATA IN from previous Arduino DATA OUT

RED - s88 pin 6 V+ - Arduino 5V

IMPORTANT: To avoid S88 signals to jitter, it is best to put DATA_in pin 12 to GND on the last Arduino in the chain.

Rudy also provides this:

Connections:

The wiring between ECoS and Arduino, looking at the S88 connector, from left to right:

S88 pin – Arduino pin

1 data – 13 (dataOut and LED indicator)

2 GND – GND

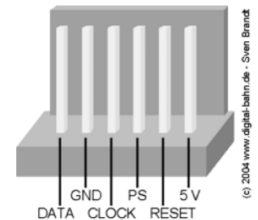
3 Clock – 2 (=interrupt 0)

4 PS – 3 (=interrupt 1)

5 Reset – not used

6 Vcc – 5V

not conn – 12 (dataIn)



The sketch on the right should be interpreted as follows: Looking at the ECoS from the back (cable connection side), the S88 connector is at the far right. **The connector latch is on the top side and the 5v pin is the far-right pin.**

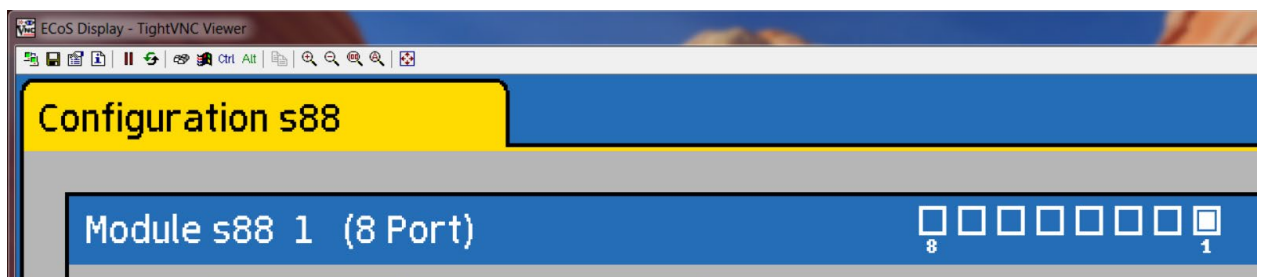
- Now build my breadboard with these 6 pins wired so that the cable can plug into the bread board and connect to the Arduino. In place of any sensor connections, wire in a tactile micro switch that will connect each of Rudy's output pins to ground. I used one switch and simply moved the pin wire from Rudy's defined pin 1 to pin 16 consecutively.
- The 16 pins are defined in Rudy's code as (pins A0 to A5; Pins 0,1,4 thru 11) follows:

```
41 pinMode(dataOut, OUTPUT); //pin 13 = data
42 digitalWrite(dataOut, LOW); //LED off
43 pinMode(A0, INPUT_PULLUP); //sensor 01
44 pinMode(A1, INPUT_PULLUP); //sensor 02
45 pinMode(A2, INPUT_PULLUP); //sensor 03
46 pinMode(A3, INPUT_PULLUP); //sensor 04
47 pinMode(A4, INPUT_PULLUP); //sensor 05
48 pinMode(A5, INPUT_PULLUP); //sensor 06
49 pinMode(0, INPUT_PULLUP); //sensor 07
50 pinMode(1, INPUT_PULLUP); //sensor 08
51 pinMode(4, INPUT_PULLUP); //sensor 09
52 pinMode(5, INPUT_PULLUP); //sensor 10
53 pinMode(6, INPUT_PULLUP); //sensor 11
54 pinMode(7, INPUT_PULLUP); //sensor 12
55 pinMode(8, INPUT_PULLUP); //sensor 13
56 pinMode(9, INPUT_PULLUP); //sensor 14
57 pinMode(10, INPUT_PULLUP); //sensor 15
58 pinMode(11, INPUT_PULLUP); //sensor 16
59 }
60
```

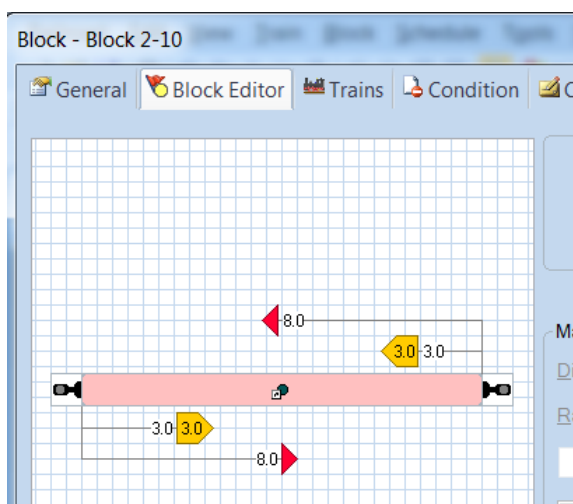
Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

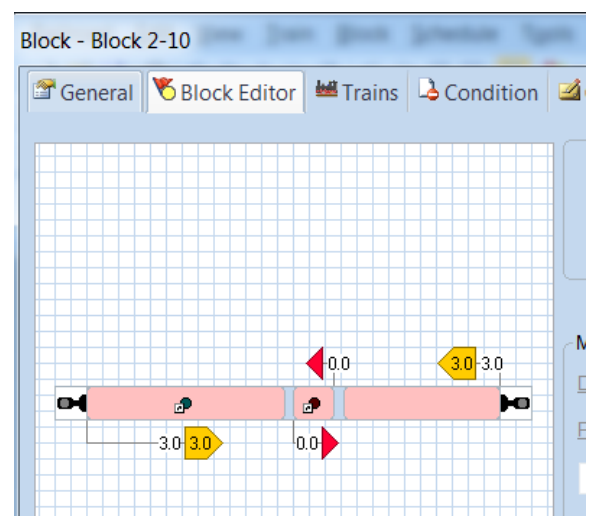
- Loaded up Rudy's S88 software to the Arduino, plugged the cable in and also into the ECoS. Ensured that I had one S88 device configured in the ECoS. I then proceeded to test each pin consecutively, using the micro switch to close the circuit and look for the corresponding indication that the ECoS was receiving the S88 signal. This all worked perfectly except for the first pin A0 which fired S88 port 1 and port 8 together for some unknown reason.
- However, in considering the next step – I realized that I needed the Analog input pins to read my LDR's and I thus did not have enough Arduino pins for a 16 port device. As the ECoS can also be setup to take input from 8 pin S88 device, I modified Rudy's code to free up all the analog pins and to start the 8-pin device at digital pin #4. After modifying the ECoS setup to an 8-pin device I retested the output of Rudy's (now modified) code and each of the 8 pins fired perfectly. Picture below shows digital pin 4 firing S88 port 1.



- The next step is to 'combine' the two bits of code together. This is the point where it got sticky. In fact, when I put the code together, I could not get it to work. The fact is that my changes are reading an analog pin to decide (in software) if the loco has arrived and then setting a digital pin, this is quite different from what Rudy was doing – simply opening a path to GND thru the digital pin via the reed switch. I called and had a really good conversation with Ross, about this issue, and left with a few directions to try. But the next morning, I re-read the Arduino documentation and realized that Rudy was using "input" pins and as I had no wires/sensors attached to my pins – I needed to set them as Digital OUTPUT pins. Making this one change – **made my code work.**
- Train controller setup: I simply changed my blocks to incorporate the new sensor and moved my stop measurement from the occupancy sensor to the S88 stop sensor:



Original Block



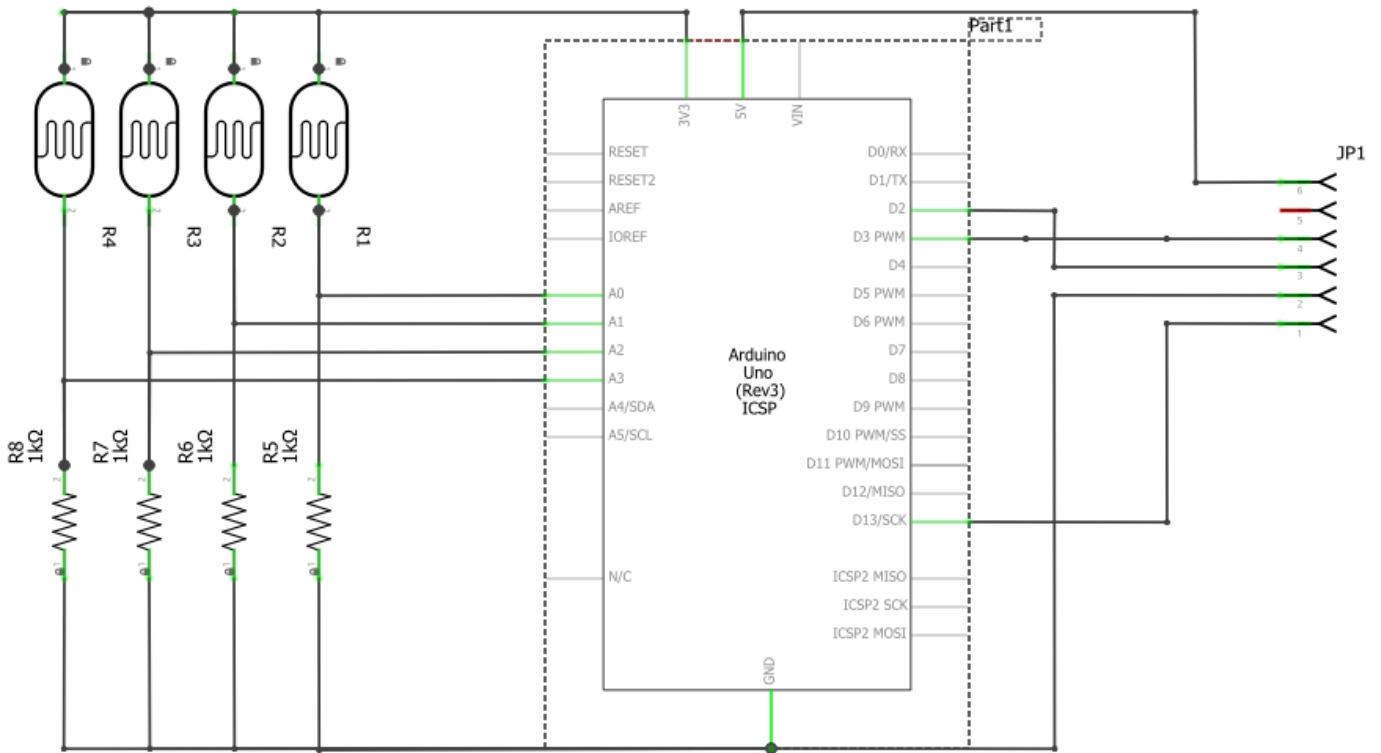
Revised Block

In this case the LHS and RHS block sensor are the same original current sensed block sensors. The centre sensor is the new S88 LDR triggered sensor for defining the STOP point.

Tip: Arduino 8 Port s88 by Mike Jago

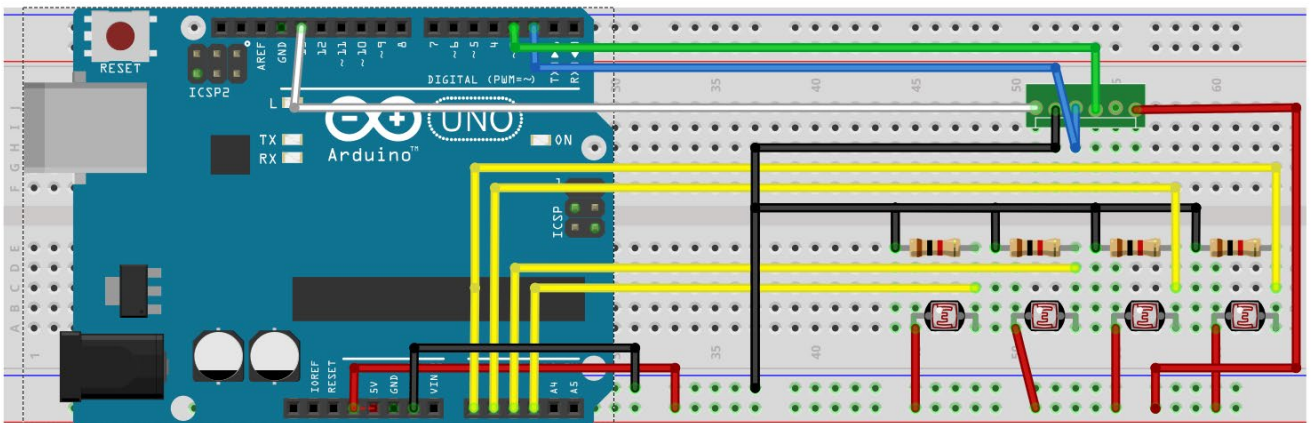
Date: 24-10-2019, New 16-01-2024

14. Fritzing Circuit Schematic:



See supplied Fritzing circuit file in the supplied files on page 10, this file can only be used with the Fritzing software.

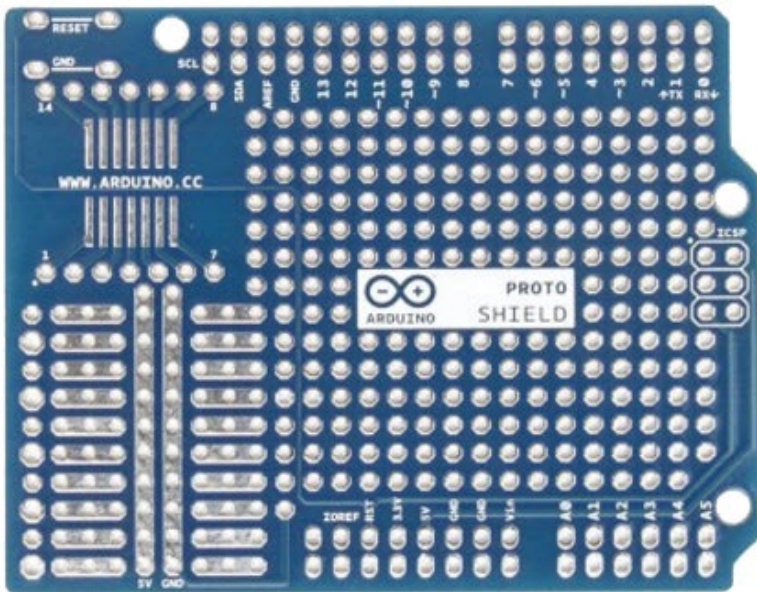
15. Fritzing Breadboard:



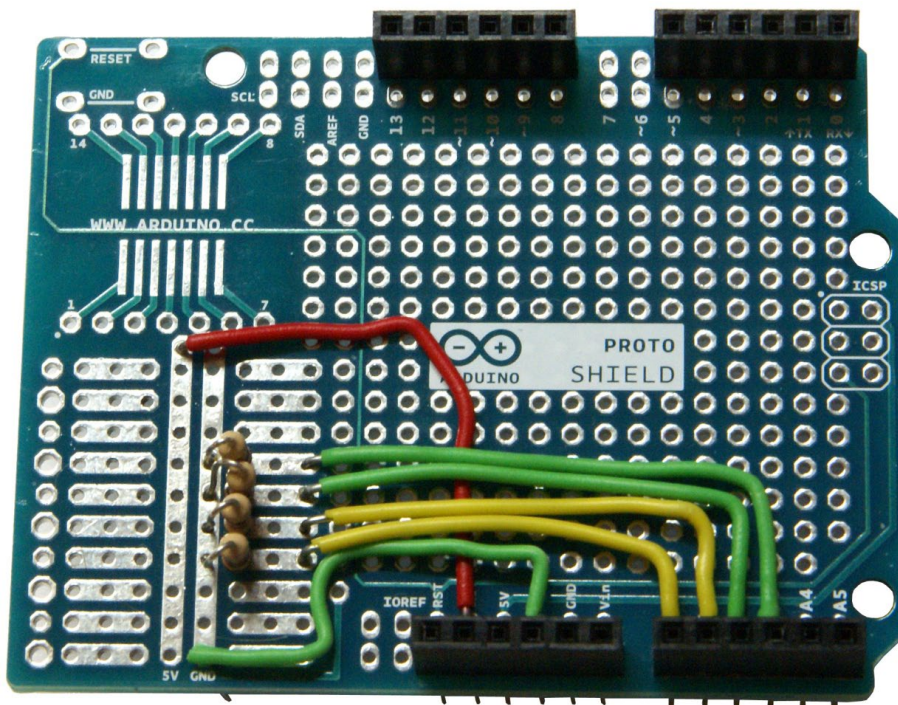
Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

16. Building the [Arduino Shield](#) shown here for reference.



- The area to the bottom left looks ideal to make connections to my LDR's
- Tack-solder the Arduino headers in place with the pins engaged in the Arduino, then remove from the Arduino and solder all pins.
- Now layout the PCB from the circuit diagram and breadboard constructed for testing purposes. Here you can see how I am planning to connect my 4 LDR's to pins A0 thru A3 and the resistors shown on the circuit diagram to the GND rail.

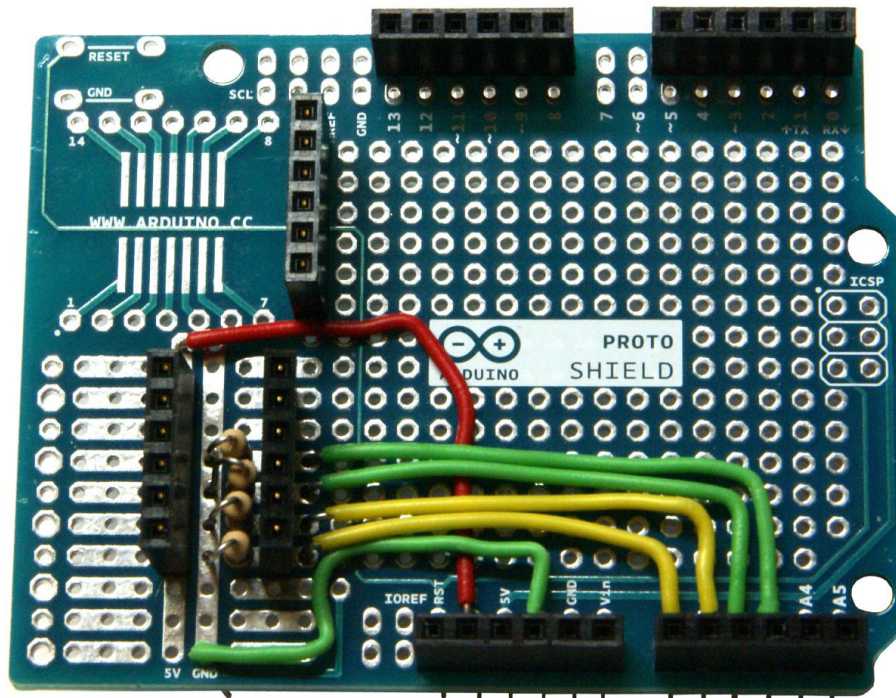


- While the shield board is marked for a +5v rail – we will use the +3.3v pin to power this rail.

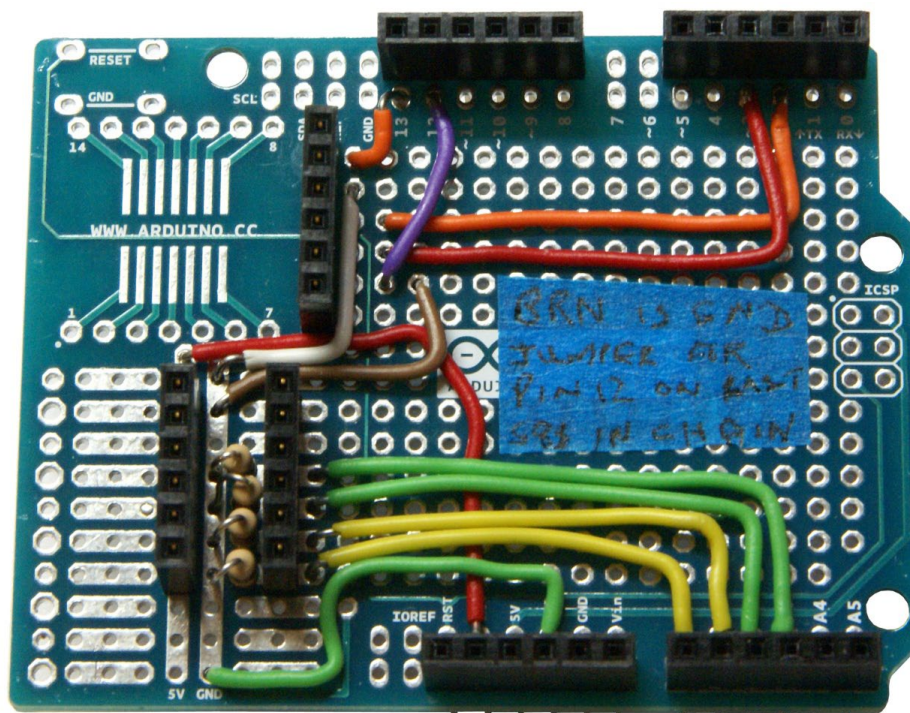
Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

- Next position the two headers for connecting the LDR sensors – one on the 3.3v DC rail and one on the ‘3 hole’ solder pad as shown. Solder these two headers in place.



- Finally position the header for the S88 cable in a convenient spot. We need to wire in pins 2, 3 and 13 to the header as well as GND. I skipped the 5v connection as I am powering the Arduino with my layout 12v power & thus not expecting to receive 5v power from the ECoS.

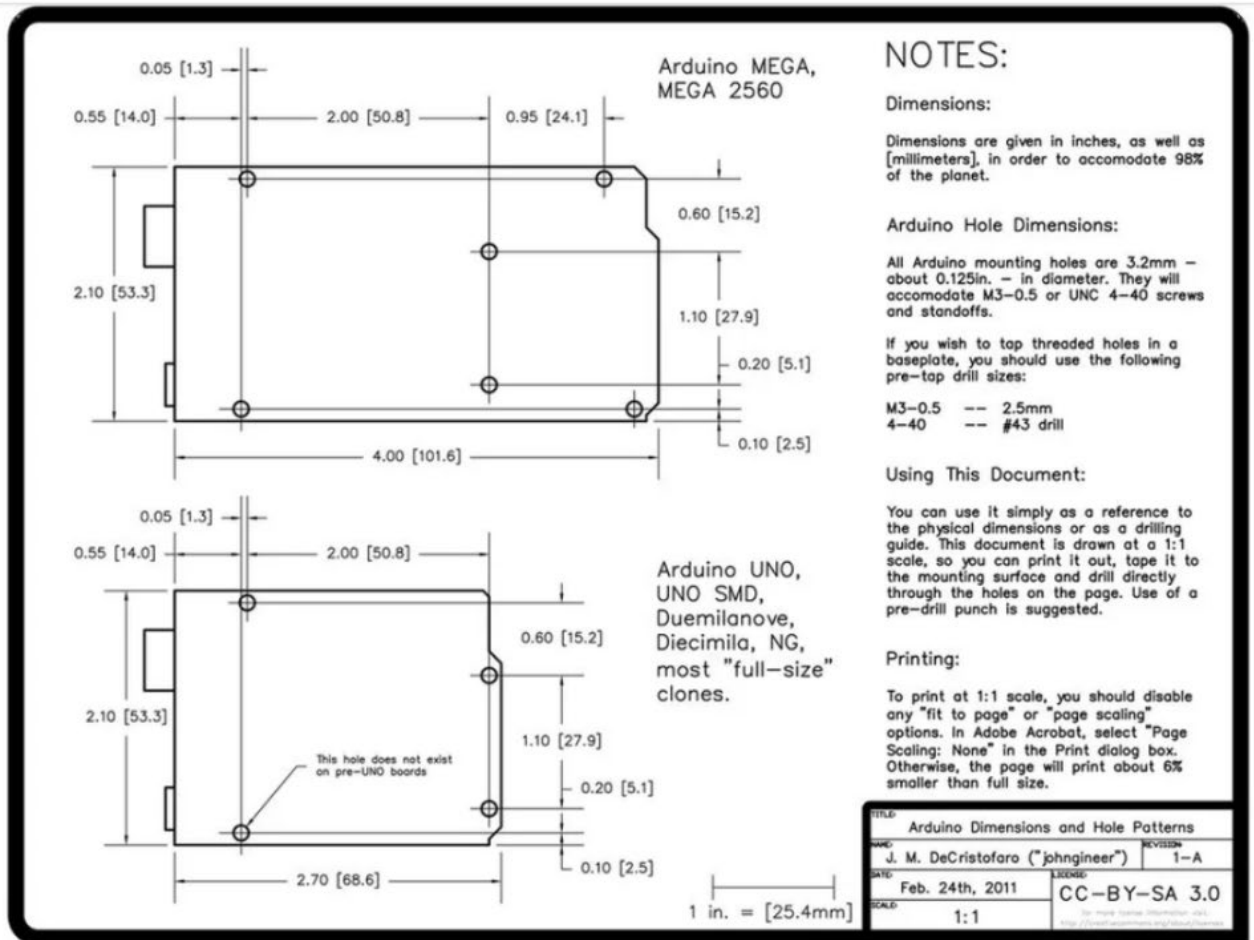


- Note the GND and Vin pins on the centre 6 pin header – I am feeding 12vDC (and GND) to these two pins to power everything. Also, you may have noted Rudy’s comment about grounding pin 12 on the last Arduino in the chain. I do that here with the **PUR** (purple) wire and my blue tape note is a reminder to me to cut this jumper if I add another Arduino into the chain.

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

- Mounting hole Dims (NTS):



I am up and running with 4 stop sensors and, so far, very happy with the result.

[Mike's supplied data files](#)

This zip file contains **RB_S88_LDR.ino** and **S88-LDR.fzz**

As always enjoy your model trains.

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

Appendix A

RB_S88_LDR.ino Mikes final Arduino file in text format

```
/*
Modified M Jagoe Oct 2019 - FINAL Oct 18, 2019
this mod to test an 8 pin S88 output, so that I have analog pins free
for sensors

S88 occupancy sensor interface to Command Station (in my case an ESU ECoS2)

Software by Ruud Boer, November 2014.
Freely distributable for private, non commercial, use.

Connections for S88 bus:
s88 pin 1 Data - Arduino pin 13 = Data_Out to Oommand Station, or to the previous
Arduino in the chain
s88 pin 2 GND - Arduino GND
s88 pin 3 Clock - Arduino pin 2, interrupt 0
s88 pin 4 PS - Arduino pin 3, interrupt 1
S66 pin 5 Reset (not used here) - Arduino pin 12, used as DATA IN from previous
Arduino DATA OUT
s88 pin 6 V+ - Arduino 5V

IMPORTANT: To avoid S88 signals to jitter, it is best to put DATA_in pin 12 to GND on
the last Arduino in the chain.

Connections for sensors: see table in void Setup() at line 35.
REMARK1: Inputs have the internal pullup resistor active, the sensors must pull the
input to GND.
REMARK2: How short a pulse is allowed from the sensors before it is not seen?
A test showed that the main loop where all sensors are read runs once every 76
microseconds.
If a train runs over the reed switch with a speed of 1m/s, which is over 300 km/hr,
that translates to 1 mm/ms.
So even if the reed switch would be on only for a 1 mm travel distance, then still
the Arduino
will read that info more than 10 times!

*/

int clockCounter=0;
long loopCounter=0; //used in lines 55 and 88, see there for explanation
unsigned int sensors=0;
unsigned int data=0xffff;
const byte dataIn=12; //data input from next Arduino in S88 chain
const byte dataOut=13; //data output pin=13
boolean loadSensors=false; //flag that says to load sensor bits into dataOut bits
// additions by MJ for LDR sensors:
int val; // read from analog pin;- the resistance

// define LDR:
//the analog pins used for each sensor
const byte SENSOR_1 = 0; // Pin A0
const byte SENSOR_2 = 1; // Pin A1
const byte SENSOR_3 = 2; // Pin A2
const byte SENSOR_4 = 3; // Pin A3

//the threshold values for each sensor
int sensor1 = 1024;
int sensor2 = 1024;
int sensor3 = 1024; //
```

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

```
int sensor4 = 1024; //

void setup() {
  Serial.begin(38400); // added for debug MJ comment out when running code
  // Calibrating LDR sensors:
  Serial.println("Calibrating Sensor" ); // calibrating sensors:
  while (millis() < 5000) {
    //for the first five seconds check and store the lowest light
    //level seen on each sensor
    sensor1 = min(sensor1, analogRead(SENSOR_1));
    sensor2 = min(sensor2, analogRead(SENSOR_2));
    sensor3 = min(sensor3, analogRead(SENSOR_3)); //not currently installed
    sensor4 = min(sensor4, analogRead(SENSOR_4)); //not currently installed
  }
  Serial.print("Sensor 1  " );
  Serial.println(sensor1);
  Serial.print("Sensor 2  " );
  Serial.println(sensor2);
  Serial.print("Sensor 3  " ); //not currently installed
  Serial.println(sensor3);
  Serial.print("Sensor 4  " ); //not currently installed
  Serial.println(sensor4);
  delay (5000);
  //the cut off level for triggering the state machine
  //is half the resistance seen during calibration
  sensor1 = sensor1 * 0.5;
  sensor2 = sensor2 * 0.5;
  sensor3 = sensor3 * 0.5;
  sensor4 = sensor4 * 0.5;
  Serial.println("Done" );
  // End LDR additions

  pinMode(2, INPUT_PULLUP);
  attachInterrupt(0,clock,RISING); //pin 2 = clock interrupt
  pinMode(3, INPUT_PULLUP);
  attachInterrupt(1,PS,RISING); //pin 3 = PS interrupt
  pinMode(dataIn,INPUT_PULLUP); //pin 12 = data in from next Arduino S88 in chain
  pinMode(dataOut, OUTPUT); //pin 13 = data out to ECoS or to previous Arduino in S88
  chain
  digitalWrite(dataOut, LOW); //LED off
  // pinMode(A0, INPUT_PULLUP); //sensor 01
  // pinMode(A1, INPUT_PULLUP); //sensor 02
  // pinMode(A2, INPUT_PULLUP); //sensor 03
  // pinMode(A3, INPUT_PULLUP); //sensor 04
  // pinMode(A4, INPUT_PULLUP); //sensor 05
  // pinMode(A5, INPUT_PULLUP); //sensor 06
  // pinMode(0, INPUT_PULLUP); //sensor 07
  // pinMode(1, INPUT_PULLUP); //sensor 08
  pinMode(4, OUTPUT); //sensor 01 was 09 ; was INPUT_PULLUP -- same change to all 8
  pins below
  pinMode(5, OUTPUT); //sensor 02 was 10
  pinMode(6, OUTPUT); //sensor 03 was 11
  pinMode(7, OUTPUT); //sensor 04 was 12
  pinMode(8, OUTPUT); //sensor 05 was 13
  pinMode(9, OUTPUT); //sensor 06 was 14
  pinMode(10, OUTPUT); //sensor 07 was 15
  pinMode(11, OUTPUT); //sensor 08 was 16
}

void loop() {
  if (loopCounter==600){bitSet(sensors,0);}
  /*
```

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

For an unknown reason the ECoS sets the first 8 bits to 1 after startup / reset of the S88 Arduino's.

When one of the sensor inputs is changed, from there on everything goes well.

Therefore, over here we give sensor bit 0 an automatic change after 30 seconds, when the ECoS is fully started.

The 1 second is created via 'loopCounter', which increments in the PS interrupt (line 88).

There are approximately 20 PS pulses per second, therefore we use $20 \times 30 = 600$ in the if statement.

```
*/

// Insert code for LDR's to read and set the respective digital pin
digitalWrite(4, HIGH); // set the pin HIGH
val = analogRead(SENSOR_1);
if (val < sensor1) {
// Serial.println("Sensor 1 triggered");
digitalWrite(4, LOW); // set the pin LOW to trigger an S88 output
}
digitalWrite(5, HIGH); // set the pin HIGH
val = analogRead(SENSOR_2);
if (val < sensor2) {
// Serial.println("Sensor 2 triggered");
digitalWrite(5, LOW); // set the pin LOW to trigger an S88 output
}
digitalWrite(6, HIGH); // set the pin HIGH
val = analogRead(SENSOR_3);
if (val < sensor3) {
Serial.println("Sensor 3 triggered");
digitalWrite(6, LOW); // set the pin LOW to trigger an S88 output
}
digitalWrite(7, HIGH); // set the pin HIGH
val = analogRead(SENSOR_4);
if (val < sensor4) {
Serial.println("Sensor 4 triggered");
digitalWrite(7, LOW); // set the pin LOW to trigger an S88 output
}
digitalWrite(8, HIGH); // set the pin HIGH
digitalWrite(9, HIGH); // set the pin HIGH
digitalWrite(10, HIGH); // set the pin HIGH
digitalWrite(11, HIGH); // set the pin HIGH
// END LDR additions
// Following code disabled as these pins are now used for LDR readings
// if (!digitalRead(A0)) {bitSet(sensors,0);Serial.println("A0 triggered");}
// if (!digitalRead(A1)) {bitSet(sensors,1);}
// if (!digitalRead(A2)) {bitSet(sensors,2);}
// if (!digitalRead(A3)) {bitSet(sensors,3);}
// if (!digitalRead(A4)) {bitSet(sensors,4);}
// if (!digitalRead(A5)) {bitSet(sensors,5);}
// if (!digitalRead(0)) {bitSet(sensors,6);}
// if (!digitalRead(1)) {bitSet(sensors,7);}
// Start of 8 port S88 output:
if (!digitalRead(4)) {bitSet(sensors,0);} // set by the results of A0
if (!digitalRead(5)) {bitSet(sensors,1);} // set by the results of A1
if (!digitalRead(6)) {bitSet(sensors,2);} // set by the results of A2
if (!digitalRead(7)) {bitSet(sensors,3);} // set by the results of A3
if (!digitalRead(8)) {bitSet(sensors,4);}
if (!digitalRead(9)) {bitSet(sensors,5);}
if (!digitalRead(10)) {bitSet(sensors,6);}
if (!digitalRead(11)) {bitSet(sensors,7);}
}

void PS() {
clockCounter=0;
```

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

```
    data=sensors;
    sensors=0;
    loopCounter++; //Increment loopCounter to create a timer. See line 55 for
    explanation.
}

void clock() {
    digitalWrite(dataOut,bitRead(data,clockCounter));
    delayMicroseconds(16); //Delay makes reading output signal from next Arduino in
    chain more reliable.
    bitWrite(data,clockCounter,digitalRead(dataIn));
    clockCounter =(clockCounter +1) % 16;
}
```

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

Appendix B

Combine the LDR software with the s88 software

LDR Test Software	S88 Software – Rudy’s stock standard software
<pre>/*Hacked again Sept 2019 for testing LDR's for Trolley stop sensors Signals – a 4 block setup turns all signal green on startup. on signal from nth Block, sets Block_n to red Block_n-1 to red, Block_n-2 to yellow and Block_n-3 to green by Mike Jagoe */ int n = 5; // block # int m = 0; //prior block # int val; // read from analog read fn // define LDR: //the analog pins used for each sensor const byte SENSOR_1 = 0; // Pin A0 const byte SENSOR_2 = 1; // Pin A1 const byte SENSOR_3 = 2; // Pin A2 const byte SENSOR_4 = 3; // Pin A3 //the threshold values for each sensor int sensor1 = 1024; int sensor2 = 1024; int sensor3 = 1024; int sensor4 = 1024; // the setup function runs once when you press reset or power the board void setup() { //configure serial communication Serial.begin(9600); Serial.println("Calibrating Sensor"); // calibrating sensors: while (millis() < 5000) { //for the first five seconds check and store the lowest light //level seen on each sensor sensor1 = min(sensor1, analogRead(SENSOR_1)); sensor2 = min(sensor2, analogRead(SENSOR_2)); //sensor3 = min(sensor3, analogRead(SENSOR_3)); //sensor4 = min(sensor4, analogRead(SENSOR_4));</pre>	<pre>/* S88 occupancy sensor interface to Command Station (in my case an ESU ECoS2) Software by Ruud Boer, November 2014. Freely distributable for private, non commercial, use. Connections for S88 bus: s88 pin 1 Data - Arduino pin 13 = Data_Out to Oommand Station, or to the previous Arduino in the chain s88 pin 2 GND - Arduino GND s88 pin 3 Clock - Arduino pin 2, interrupt 0 s88 pin 4 PS - Arduino pin 3, interrupt 1 S66 pin 5 Reset (not used here) - Arduino pin 12, used as DATA IN from previous Arduino DATA OUT s88 pin 6 V+ - Arduino 5V IMPORTANT: To avoid S88 signals to jitter, it is best to put DATA_in pin 12 to GND on the last Arduino in the chain. Connections for sensors: see table in void Setup() at line 35. REMARK1: Inputs have the internal pullup resistor active, the sensors must pull the input to GND. REMARK2: How short a pulse is allowed from the sensors before it is not seen? A test showed that the main loop where all sensors are read runs once every 76 microseconds. If a train runs over the reed switch with a speed of 1m/s, which is over 300 km/hr, that translates to 1 mm/ms. So even if the reed switch would be on only for a 1 mm travel distance, then still the Arduino will read that info more than 10 times! */ int clockCounter=0; long loopCounter=0; //used in lines 55 and 88, see there for explanation unsigned int sensors=0; unsigned int data=0xffff;</pre>

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

<pre>} Serial.print("Sensor 1 "); Serial.println(sensor1); Serial.print("Sensor 2 "); Serial.println(sensor2); Serial.print("Sensor 3 "); Serial.println(sensor3); Serial.print("Sensor 4 "); Serial.println(sensor4); delay (5000); //the cut off level for triggering the state machine //is half the resistance seen during calibration sensor1 = sensor1 * 0.5; sensor2 = sensor2 * 0.5; sensor3 = sensor3 * 0.5; sensor4 = sensor4 * 0.5; Serial.println("Done"); } // the loop function runs over and over again forever void loop() { val = analogRead(SENSOR_1); n=0; if (val < sensor1) { Serial.print("Sensor 1 triggered"); Serial.print(val); Serial.print(" "); Serial.println(n); } val = analogRead(SENSOR_2); n=1; if (val < sensor2) { Serial.print("Sensor 2 triggered"); Serial.print(val); Serial.print(" "); Serial.println(n); } val = analogRead(SENSOR_3); n=2; if (val < sensor3) { Serial.print("Sensor 3 triggered"); Serial.print(val); Serial.print(" "); Serial.println(n); } }</pre>	<pre>const byte dataIn=12; //data input from next Arduino in S88 chain const byte dataOut=13; //data output pin=13 boolean loadSensors=false; //flag that says to load sensor bits into dataOut bits void setup() { Serial.begin(38400); // added for debug MJ comment out when running pinMode(2, INPUT_PULLUP); attachInterrupt(0,clock,RISING); //pin 2 = clock interrupt pinMode(3, INPUT_PULLUP); attachInterrupt(1,PS,RISING); //pin 3 = PS interrupt pinMode(dataIn,INPUT_PULLUP); //pin 12 = data in from next Arduino S88 in chain pinMode(dataOut, OUTPUT); //pin 13 = data out to ECoS or to previous Arduino in S88 chain digitalWrite(dataOut, LOW); //LED off pinMode(A0, INPUT_PULLUP); //sensor 01 pinMode(A1, INPUT_PULLUP); //sensor 02 pinMode(A2, INPUT_PULLUP); //sensor 03 pinMode(A3, INPUT_PULLUP); //sensor 04 pinMode(A4, INPUT_PULLUP); //sensor 05 pinMode(A5, INPUT_PULLUP); //sensor 06 pinMode(0, INPUT_PULLUP); //sensor 07 pinMode(1, INPUT_PULLUP); //sensor 08 pinMode(4, INPUT_PULLUP); //sensor 09 pinMode(5, INPUT_PULLUP); //sensor 10 pinMode(6, INPUT_PULLUP); //sensor 11 pinMode(7, INPUT_PULLUP); //sensor 12 pinMode(8, INPUT_PULLUP); //sensor 13 pinMode(9, INPUT_PULLUP); //sensor 14 pinMode(10, INPUT_PULLUP); //sensor 15 pinMode(11, INPUT_PULLUP); //sensor 16 } void loop() { if (loopCounter==600){bitSet(sensors,0);} /* For an unknown reason the ECoS sets the first 8 bits to 1 after startup / reset of the S88 Arduino's. When one of the sensor inputs is changed, from there on everything goes well. Therefore, over here we give sensor bit 0 an automatic change after 30 seconds, when the ECoS is fully started. The 1 second is created via 'loopCounter', which increments in the PS interrupt (line 88). There are approximately 20 PS pulses per second, therefore we use 20x30=600 in the if statement.</pre>
--	--

Tip: Arduino 8 Port s88 by Mike Jagoe

Date: 24-10-2019, New 16-01-2024

<pre>val = analogRead(SENSOR_4); n=3; if (val < sensor4) { Serial.print("Sensor 4 triggered"); Serial.print(val); Serial.print(" "); Serial.println(n); } }</pre>	<pre>*/ if (!digitalRead(A0)) {bitSet(sensors,0);Serial.println("A0 triggered");} if (!digitalRead(A1)) {bitSet(sensors,1);} if (!digitalRead(A2)) {bitSet(sensors,2);} if (!digitalRead(A3)) {bitSet(sensors,3);} if (!digitalRead(A4)) {bitSet(sensors,4);} if (!digitalRead(A5)) {bitSet(sensors,5);} if (!digitalRead(0)) {bitSet(sensors,6);} if (!digitalRead(1)) {bitSet(sensors,7);} if (!digitalRead(4)) {bitSet(sensors,8);} if (!digitalRead(5)) {bitSet(sensors,9);} if (!digitalRead(6)) {bitSet(sensors,10);} if (!digitalRead(7)) {bitSet(sensors,11);} if (!digitalRead(8)) {bitSet(sensors,12);} if (!digitalRead(9)) {bitSet(sensors,13);} if (!digitalRead(10)) {bitSet(sensors,14);} if (!digitalRead(11)) {bitSet(sensors,15);} } void PS() { clockCounter=0; data=sensors; sensors=0; loopCounter++; //Increment loopCounter to create a timer. See line 55 for explanation. } void clock() { digitalWrite(dataOut,digitalRead(data,clockCounter)); delayMicroseconds(16); //Delay makes reading output signal from next Arduino in chain more reliable. digitalWrite(data,clockCounter,digitalRead(dataIn)); clockCounter =(clockCounter +1) % 16; }</pre>
--	---