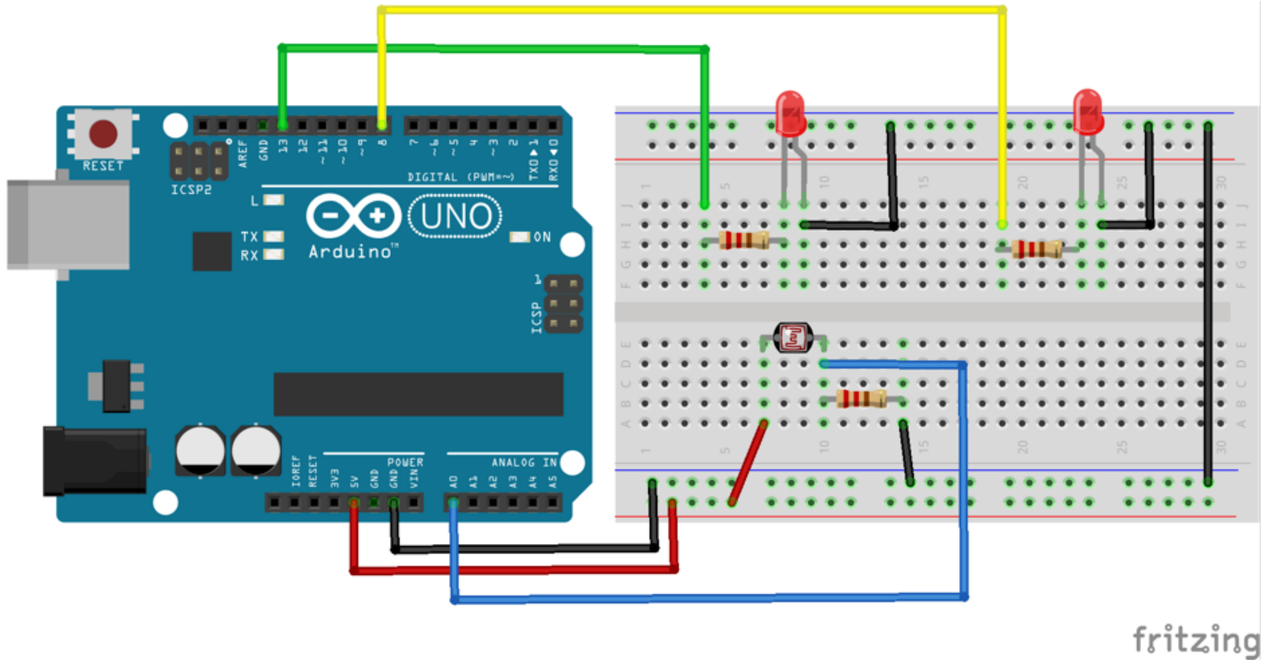
CENTER FOR URBAN SCIENCE+PROGRESS

Rapid Prototyping of Urban Sensors

Calibration Lab

CALIBRATION

- set the maximum and minimum values in the first five seconds




```
// variables:
int sensorValue = 0;         // the sensor value
int sensorMin = 1023;        // minimum sensor value
int sensorMax = 0;           // maximum sensor value

// calibrate during the first five seconds
while (millis() < 5000) {
  sensorValue = analogRead(A0);

  // record the maximum sensor value
  if (sensorValue > sensorMax) {
    sensorMax = sensorValue;
  }

  // record the minimum sensor value
  if (sensorValue < sensorMin) {
    sensorMin = sensorValue;
  }
}

// apply the calibration to the sensor reading
sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);

// in case the sensor value is outside the range seen during calibra
sensorValue = constrain(sensorValue, 0, 255);
```

THRESHOLD

- trigger something when the input goes above a certain value

```
int threshold = 200;    // an arbitrary threshold value


if( sensorValue < threshold){
   digitalWrite(8, HIGH);
} else {
   digitalWrite(8, LOW);
}
```

SMOOTHING

- store values ten values in an array and average with each new incoming value

```
const int numReadings = 20;

int readings[numReadings];  // the readings from the analog input
int readIndex = 0;          // the index of the current reading
int total = 0;              // the running total
int average = 0;            // the average
```

```
for (int thisReading = 0; thisReading < numReadings; thisReading++){
  readings[thisReading] = 0;
}
```

```
// subtract the last reading
total = total - readings[readIndex];
// read from the sensor
readings[readIndex] = sensorValue;
// add the reading to the total
total = total + readings[readIndex];
// advance to the next positioon in the array
readIndex = readIndex + 1;

// if we're at the end of the array...
if(readIndex >= numReadings){
   // ... wrap around to the beginning
   readIndex = 0;
}

// calculat the average
average = total / numReadings;
delay(1); // delay in between reads for stability
```