NELSON BAMFORD
SZE "RON" CHAU
ARIEL MARTINEZ

ENGINEERING CALCULATIONS

–for–

a Home Surveillance System

WENTWORTH INSTITUTE OF TECHNOLOGY
ELEC 4500: SENIOR ELECTRONIC DESIGN I

INSTRUCTOR: PROFESSOR GRENQUIST
SPRING 2016

Abstract

Technical specifications, i.e. calculations, breadboard diagrams, schematics, PCB layout design, and program flowcharts for a Raspberry Pi 2-based home surveillance system is outlined in this document. This system is designed to allow a homeowner to record digital video–through low-ambient lighting environments–when an adjustable motion threshold is surpassed. The system will include human and vehicle detection features. Recorded videos will be cataloged by the object type that activated the camera, i.e. human, vehicle, and others. This proposed functionality will allow a homeowner to quickly access their database of video data through categorical "bookmarks" based on metadata such as date, time, and type of object. The result is a lightweight, customizable, and scalable home surveillance system that remains inexpensive for storage.

Contents

List of Figures

List of Tables

1 Calculations

1.1 Storage

Storage requirements for 24 hours of 720p video data (.mp4):

$$\begin{aligned} Continual &= \left(\frac{24\ hr}{1\ day}\right)\left(\frac{60\ min}{1\ hr}\right)\left(\frac{60\ sec}{1\ min}\right)\left(\frac{17\ mbit}{1\ sec}\right)\left(\frac{1\ byte}{8\ bit}\right)\left(\frac{1\ GB}{1737441824\ byte}\right) \\ &= 170.99\ \frac{GB}{day} \end{aligned}$$

By limiting the active recording to moments where objects are detected, this storage requirement can be significantly lowered. Taking a theoretical family of 4 and a single dog, we can calculate for a minimum duration that the RPi2 is expected to record in 24 hours:

Table 1: Known sighting during an average 24 hour day for a front door camera.

Type	Qty	Duration (min)	Sighting/24 hr	Total
Person	4	1	2	8
Dog	1	1	2	2
Delivery	2	1	1	2
				12 min

With a minimum recording duration of 12 minutes based on the expected sightings (detected objects), the floor data requirements per 24 hours becomes:

$$\begin{aligned} Floor\ &= \left(\frac{12\ min}{1\ day}\right)\left(\frac{60\ sec}{1\ min}\right)\left(\frac{17\ Mbit}{1\ sec}\right)\left(\frac{1\ byte}{8\ bit}\right)\left(\frac{1\ GB}{1737441824\ byte}\right) \\ &= 0.000014254\ \frac{GB}{day} \Rightarrow 0.014254\ \frac{MB}{day} \Rightarrow 14.254\ \frac{KB}{day} \end{aligned}$$

1.2 Night Vision

1.2.1 Analog Light Sensor

$$\log(I_O) = mx + b$$

$$I_O = m\log(x) + b$$

$$I_O = 10\log(E_v)\,[l_x]$$

The sensor has an output current range of $0\mu A$–$50\mu A$ between 1 lux–100000 lux. The breakout board converts this current into a voltage using a $68K\Omega$ resistor, giving a voltage output of $0V$–$3.4V$. Given the use-case of the light sensor, the $3.4V$ can be approximated to $3.3V$, which is a suitable voltage reference for the ATtiny85 microcontroller. With the output current range approximated to a voltage range of $0V$–$3.3V$ between 1 lux–100000 lux,

$$log_i = \log(l_{x\ MAX} - l_{x\ MIN})$$

$$= \log(100000) = 5$$

To convert the analog value to a digital value, an 8-bit ADC is used:

$$ADC = \frac{log_i}{1024\ steps} = 0.00488$$

Finally, the formula to obtain the final lux output, l_{xo}, on the RPi2:

$$l_{xo} = 10^{raw_i \times ADC} = 10^{raw_i \times 0.00488}$$

where raw_i is the raw value received by one of the analog pins on the ATtiny85.

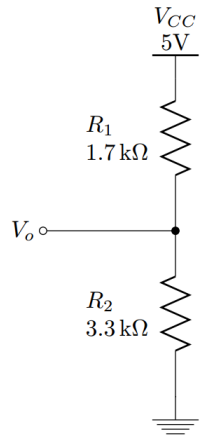


Figure 1: Voltage divider for sensor voltage reference.

Since 3.3V was chosen as the voltage reference value for the sensor, a voltage divider will be constructed in order to drop the 5V supply:

$$V_1 = V_{CC}\frac{R_2}{R_1+R_2} = 5V\frac{3.3K\Omega}{1.7K\Omega+3.3K\Omega} = 3.3V$$

1.2.2 Driving the LEDs

Single LED Iteration The 8mm IR LEDs have a forward voltage drop of 1.3V and a current recommendation of 100mA. A resistor is added in series to each IR LED with the values chosen by:

$$R_{IR} = \frac{V_{CC}-V_D}{I_D} = \frac{5V-1.3V}{100mA} = 37\Omega \approx 39\Omega$$

$$P_D = I_D^2 R_{IR} = (100mA)^2(39\Omega) = 0.39W$$

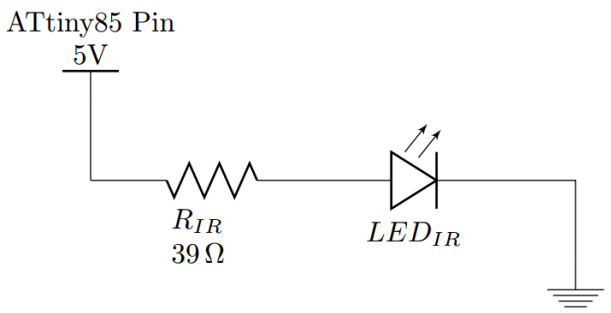


Figure 2: Single LED circuit branch on the Night Vision block.

Series LED Iteration Another configuration that is being considered includes an additional IR LED onto each of the 3 ATtiny85 output pins. This changes the series resistor value to:

$$R_{IR} = \frac{V_{CC} - V_D - V_D}{I_D} = \frac{5V - 1.3V - 1.3V}{100mA} = 24\Omega \approx 27\Omega$$

$$P_D = I_D^2 R_{IR} = (100mA)^2 (27\Omega) = 0.27W$$

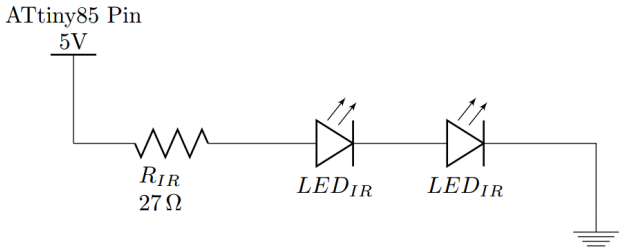


Figure 3: Double LED circuit branch on the Night Vision block.

1.3 Computer Vision: Pre-processing

Before performing feature extraction, a given image–also known as a frame–is pre-processed in order to increase the accuracy of object-class detection. Some of these pre-processing steps also improve feature extraction during low-ambient light conditions.

Some of these image processing techniques considers a given image in sections. This type of sectioning results in kernels, which is $m \times n$ matrix "windows" of a given frame. Convolution is then performed on the matrix, which averages all of the pixels around the center-most pixel.

(a) Original image. (b) Converted to grayscale.

Figure 4: An original image is converted to grayscale. A 3×3 matrix is then sectioned off for further pre-processing. A grid is used in (b) to show individual pixel boundaries.

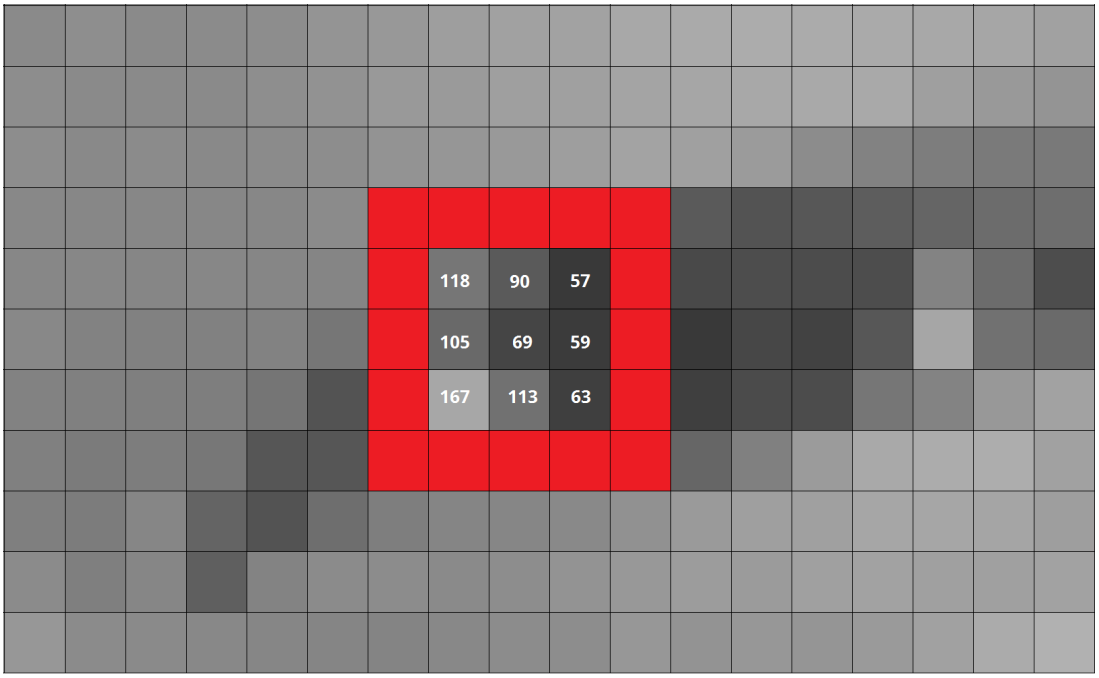


Figure 5: The values for each pixel within the 3×3 matrix.

As an arbitrary example, a 3×3 matrix in Fig. 4b, bordered in red, will be considered for calculations.

1.3.1 Gaussian Blurring

Blurring–also known as smoothing–is an image processing technique where high frequency noise is removed from a given image at the cost of detail reduction. For the purposes of this system, blurring will improve edge detection [1, p. 28, 32].

Gaussian blurring was chosen due to its ability to utilize a weighted mean when averaging pixels in each frame. This specific blurring technique will allow for better performance when thresholding the frames. The name for this blurring technique comes from its use of a Gaussian function for calculations.

$$G(x,y) = \frac{1}{\sqrt{2\pi\delta}} e^{-\frac{x^2+y^2}{2\delta^2}} \tag{1}$$

Using the example arbitrary window in the previous section, each pixel is assigned a coordinate:

$$\begin{bmatrix} (-1,1) & (0,1) & (1,1) \\ (-1,0) & (0,0) & (1,0) \\ (-1,-1) & (0,-1) & (1,-1) \end{bmatrix}$$

Choosing an arbitrary δ value of 1.5 the center coordinate is calculated through the Gaussian function:

$$G(0,0) = \frac{1}{\sqrt{2\pi 1.5}} e^{-\frac{0^2+0^2}{2\times 1.5^2}} = 0.0707$$

Calculating the rest of the coordinates results in a weighted matrix:

$$\begin{bmatrix} 0.0454 & 0.0566 & 0.0454 \\ 0.0566 & 0.0707 & 0.0566 \\ 0.0454 & 0.0566 & 0.0454 \end{bmatrix}$$

The sum of those values becomes:

$$(0.0435 \times 4) + (0.0566 \times 4) + 0.0707 = 0.4787$$

Dividing by the summation will result in a weighted average matrix:

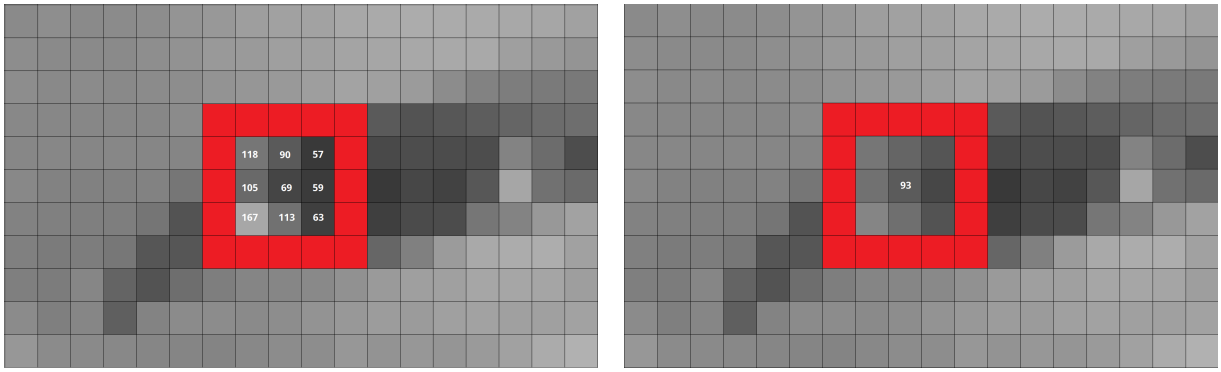
$$\frac{1}{0.479} \begin{bmatrix} 0.0454 & 0.0566 & 0.0454 \\ 0.0566 & 0.0707 & 0.0566 \\ 0.0454 & 0.0566 & 0.0454 \end{bmatrix} = \begin{bmatrix} 0.0948 & 0.118 & 0.0948 \\ 0.118 & 0.148 & 0.118 \\ 0.0948 & 0.118 & 0.0948 \end{bmatrix}$$

Multiplying each coordinate of the 3×3 window will result in a Gaussian blur:

$$\begin{bmatrix} (118 \times 0.0948) & (90 \times 0.118) & (57 \times 0.0948) \\ (105 \times 0.118) & (69 \times 0.148) & (59 \times 0.118) \\ (167 \times 0.0948) & (113 \times 0.118) & (63 \times 0.0948) \end{bmatrix} = \begin{bmatrix} 11.19 & 10.62 & 5.403 \\ 12.39 & 10.21 & 6.962 \\ 15.83 & 13.33 & 5.972 \end{bmatrix}$$

Finally, summing up the values from the last matrix will result in the new value for the center-most pixel.

$$11.19 + 10.62 + 5.403 + 12.39 + 10.21 + 6.962 + 15.83 + 13.33 + 5.972 = 91.907$$



(a) Center pixel before Gaussian blur: 69. (b) Center pixel after Gaussian blur: 93.

Figure 6: Comparison of pixel values between the original 3×3 window against one filtered with Gaussian blur.

As Fig. 6b shows, the resulting pixel value from a Gaussian blur performed in GNU Image Manipulation Program (GIMP) is within range of the calculated value.

$$\frac{|\,GIMP - Calculated\,|}{Calculated} \times 100\% = \frac{|\,93 - 91.907\,|}{91.907} \times 100\% = 0.0119\%$$

1.4 Computer Vision: Detection

Facial detection is performed by searching a given image for features based on black and white rectangles [2]. The calculation for the regions again uses matrix convolution.

The first step is a single pass through the entirety of a given image that computes the sum of all pixel values above and to the left $(+y, -x)$ of the position being examined, inclusive. This is known as an integral table, that is:

$$I_{\Sigma}(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x',y') \tag{2}$$

Using the original window in Fig. 5:

$$\begin{bmatrix} (i_{(-1,1)} = 118) & (i_{(0,1)} = 208) & (i_{(1,1)} = 265) \\ (i_{(-1,0)} = 223) & (\ddots) & (\vdots) \\ (i_{(-1,-1)} = 390) & (\cdots) & (i_{(1,-1)} = 841) \end{bmatrix}$$

Feature identification is then performed using black and white rectangles over the integral table. The algorithm then passes feature types (various rectangles created by training the classifier using positive and negative results) over the integral table [3]. Identifying features from this point on is based on a simple summation of the rectangle corners:

$$A_M = I(A) + I(C) - I(B) - I(D) \tag{3}$$

1.4.1 Haar

The robustness of this method of object detection will be apparent with this final step. Consider Fig. 7b.

(a) Haar line feature being applied to Fig. 4b. Points A to D corresponds to the white rectangle's corners.

(b) Computing the leftmost white rectangle of Fig. 4b's integral image can be quickly achieved, allowing for real-time detection.

Figure 7: Facial detection through Haar-like features.

By analyzing the integral table of the original image, feature extraction becomes com-

putationally inexpensive. If a 3×3 rectangle is covering the integral table that was solved earlier, the resulting calculation is:

$$A_M = I(A) + I(C) - I(B) - I(D) = 118 + 841 - 265 - 390 = 304$$

This value is finally compared against the collection of rectangular Haar features collected during classifier training.

Haar-like algorithm is grounded upon machine learning, a database of images containing positive and negative results for vehicles [5]. Since the detection algorithm based on Haar features is scale invariant, a set of images can then be used to train a Haar classifier for vehicles [4]. This eliminates the use of blob analysis for vehicles.

2 Breadboard Prototyping Iterations

Due to anticipated difficulties in driving the IR LEDs, there are several iterations of design choices for the system.

2.1 RPi2 GPIO-based Design

In this design, an analog-to-digital (ADC) integrated circuit is added in order to transfer the data from the analog sensor to the RPi2.

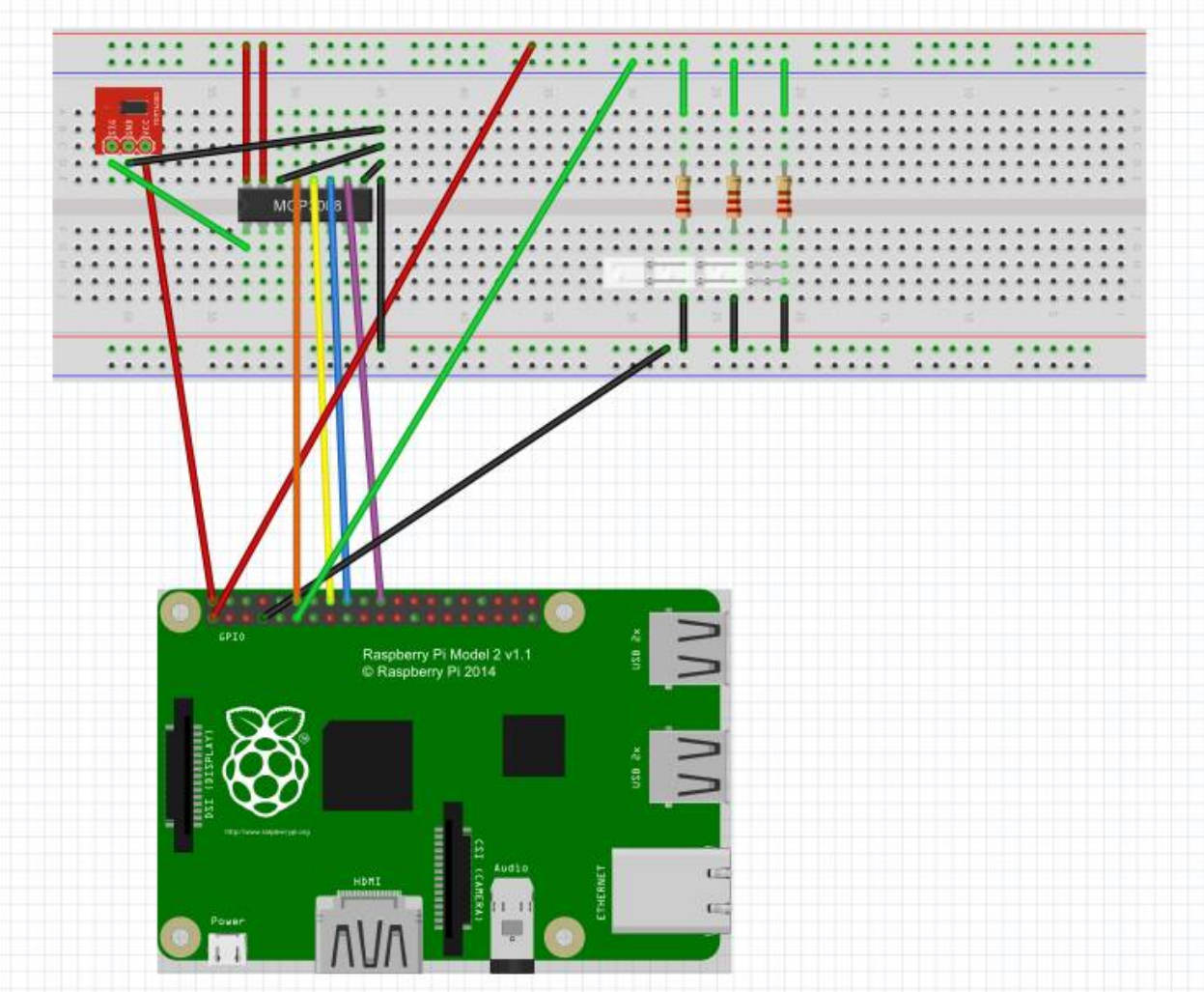
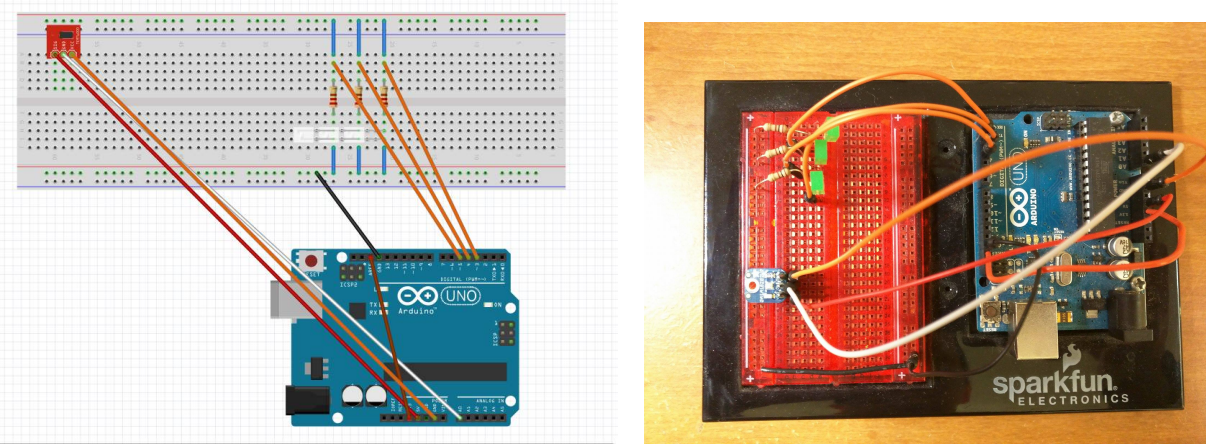


Figure 8: Fritzing breadboard diagram of the RPi2 design.

2.2 Arduino-based Design

The “Arduino-based design”, in actuality, would be an Atmel AVR series microcontroller. In this design, the Arduino Uno Rev. 3–based on the ATmega328–is utilized as a quick evaluation on whether hosting the LEDs on a microcontroller (and subsequent 5V source) would better serve the purposes of night illumination for the camera.



(a) Fritzing breadboard diagram. (b) Physical breadboard prototype.

Figure 9: Prototyping the Night Vision block on an Arduino Uno Rev. 3.

2.3 ATtiny85-based Design

The latest iteration takes the Arduino-based design and ports it over to an ATtiny85 microcontroller. Manufacturing costing for 10,000 units is the primary rationale for this decision. Since the microcontroller would only be responsible for the light sensor and the LEDs, a microcontroller from the megaAVR family would be underutilized whereas one from the tinyAVR family contains all that is necessary to host the Night Vision block.

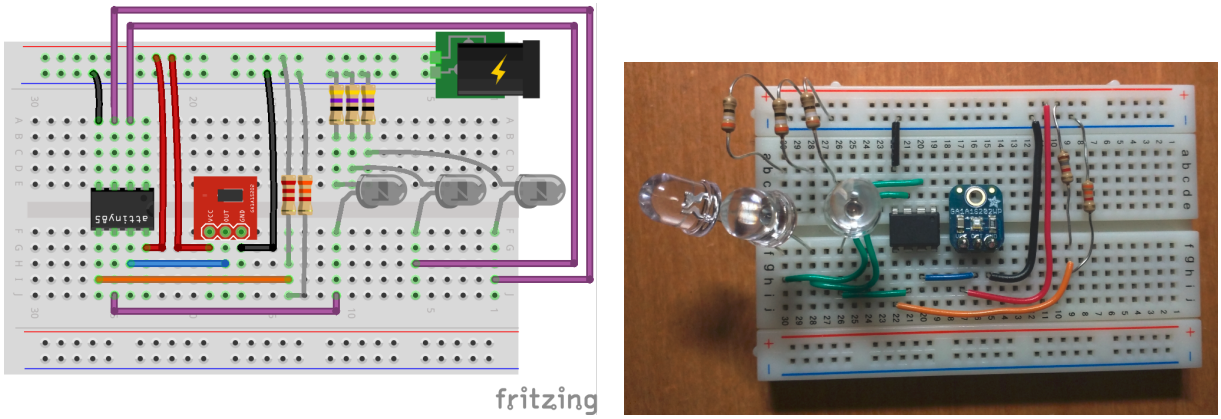
Table 2: Comparison between microcontroller candidates for the Night Vision block.

Microcontroller	Program Space	I/O Pins	Price Per Unit for 10K Units
ATmega328	32K	23	1.670
ATtiny85	8K	6	0.748

For the purposes of a manufacturing run, the cost of an ATtiny85 is also significantly less than utilizing an ADC IC to connect with the RPi2. With a design based on the ATtiny85, the RPi2 can focus solely on processing computer vision tasks.

Table 3: Comparison between ADC IC candidates for the Night Vision block.

ADC	Resolution	Channels	Price Per Unit for 10K Units
MCP3008	10-bit	8	1.660
MCP3002	10-bit	2	1.350



(a) Fritzing breadboard diagram.

(b) Physical breadboard prototype.

Figure 10: Prototyping the Night Vision block on an ATtiny85, first iteration.

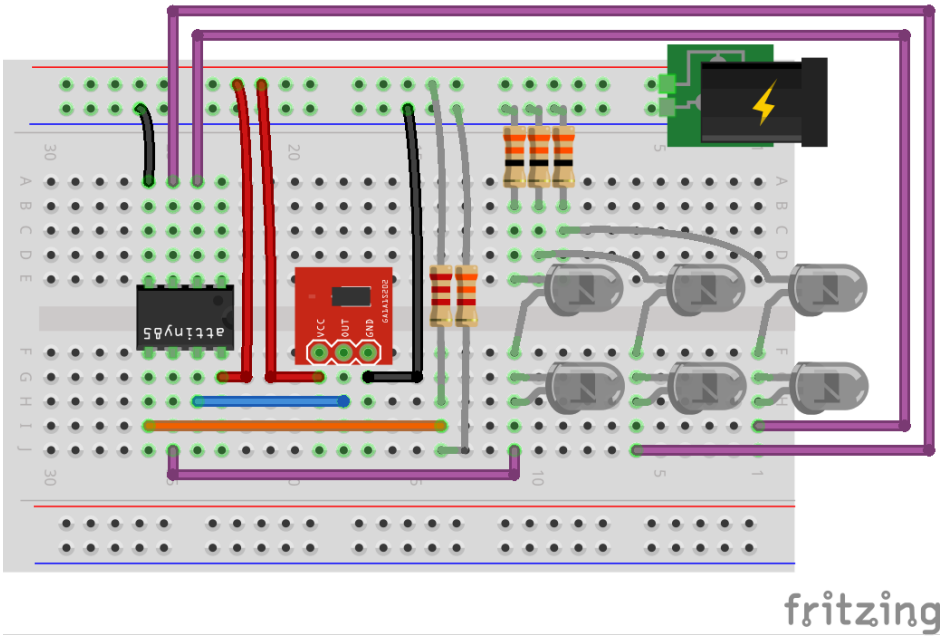


Figure 11: Fritzing breadboard diagram of the ATtiny85 design, second iteration.

3 Schematics

3.1 ATtiny85 Design: First Iteration

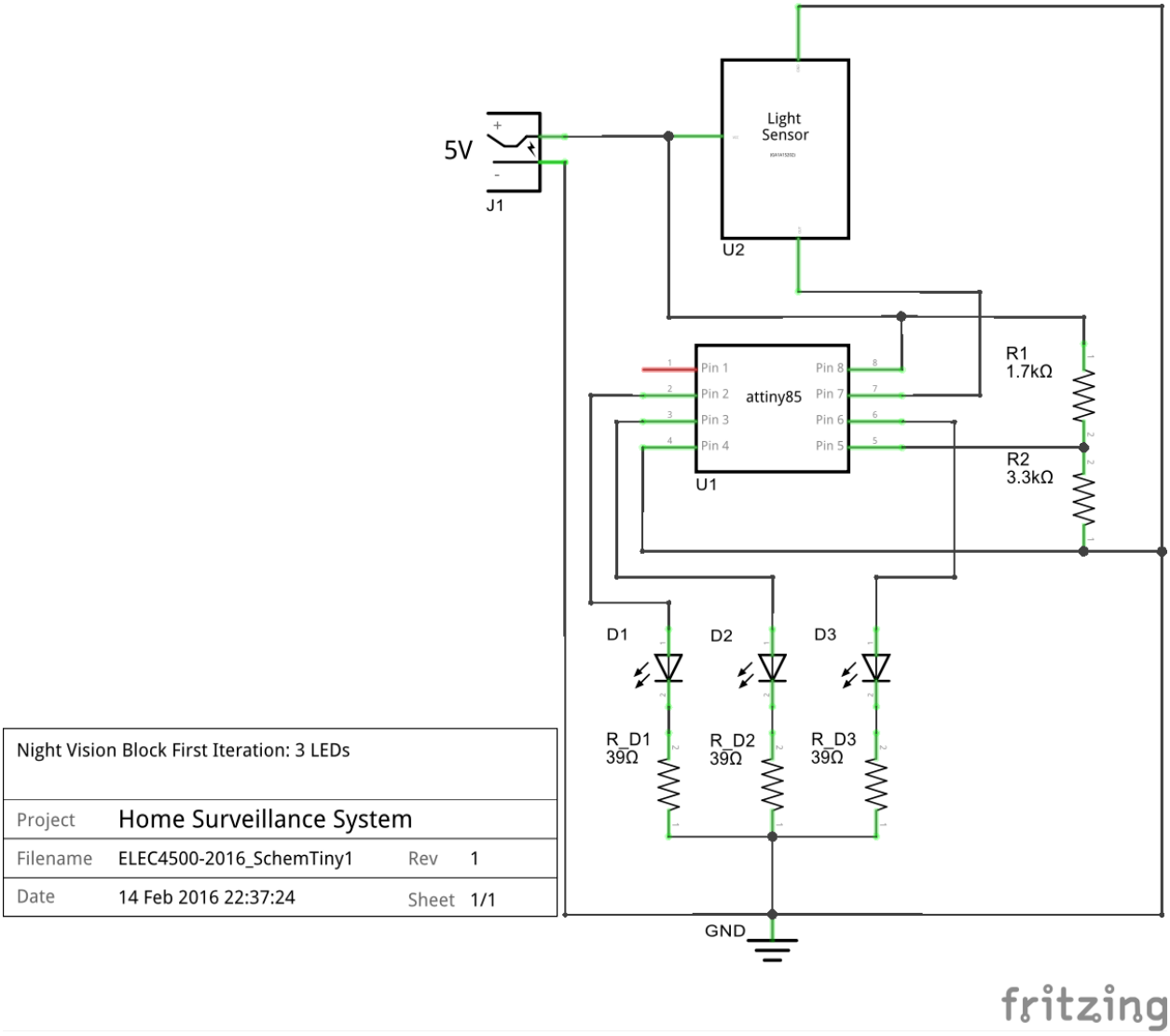


Figure 12: Fritzing schematic of the first ATtiny85 design.

3.2 ATtiny85 Design: Second Iteration

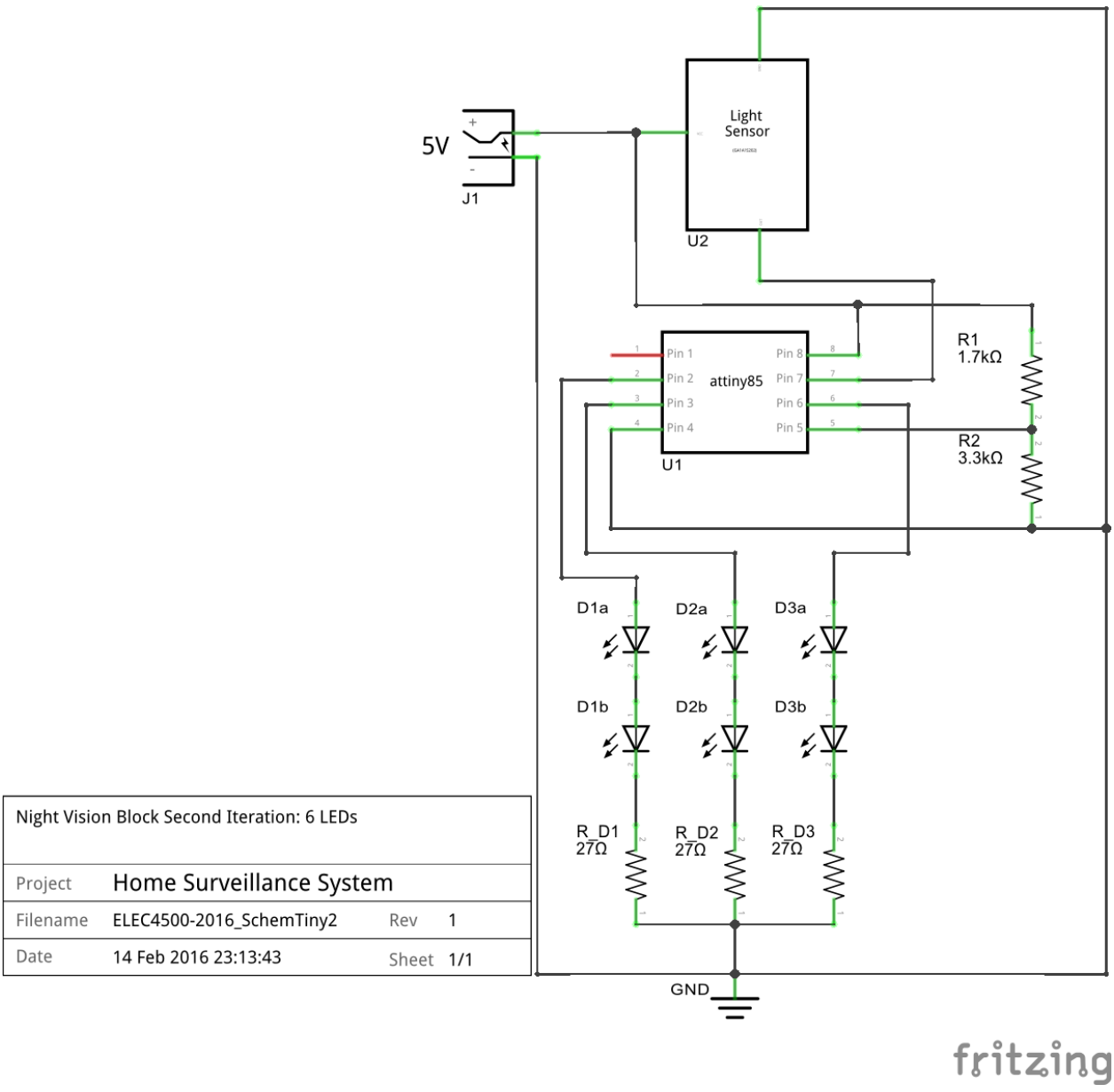


Figure 13: Fritzing schematic of the second ATtiny85 design.

4 Flowcharts

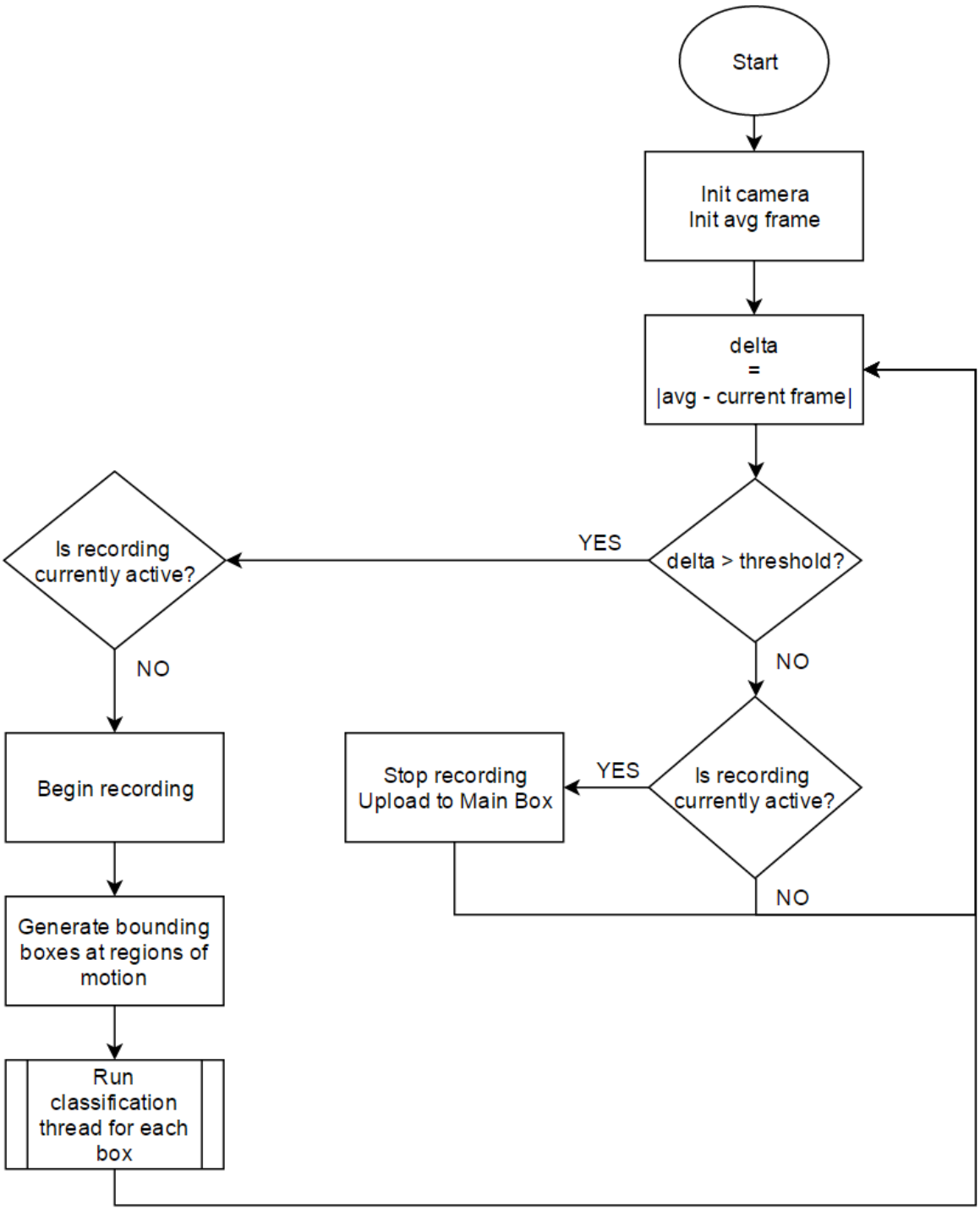


Figure 14: Flowchart of main program.

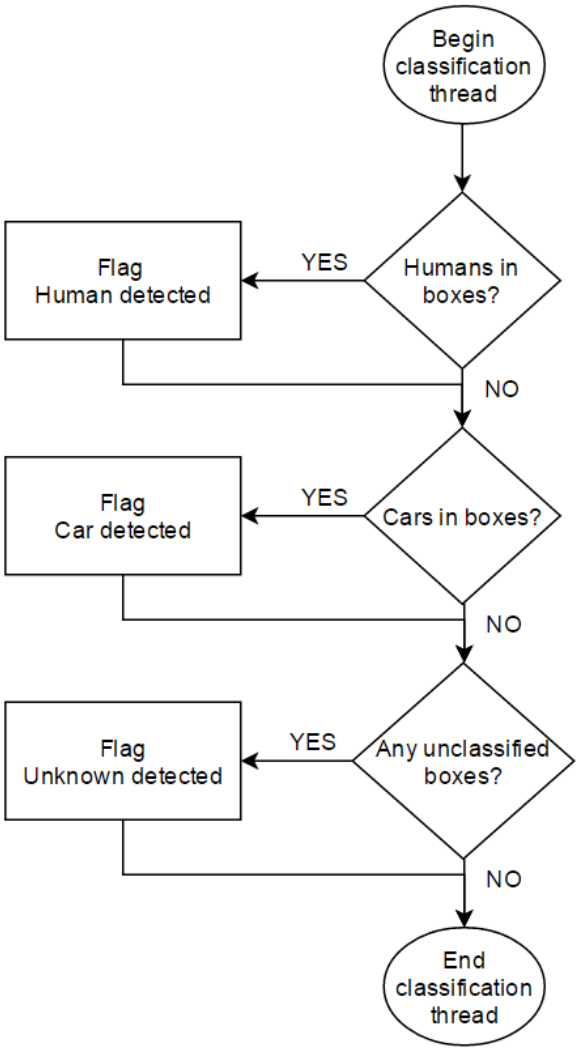


Figure 15: Flowchart of classification thread.

5 Project Development: Manufacturing

The PCB layouts as shown below are currently being produced by Fritzing Fab. The estimated delivery date is mid-March.

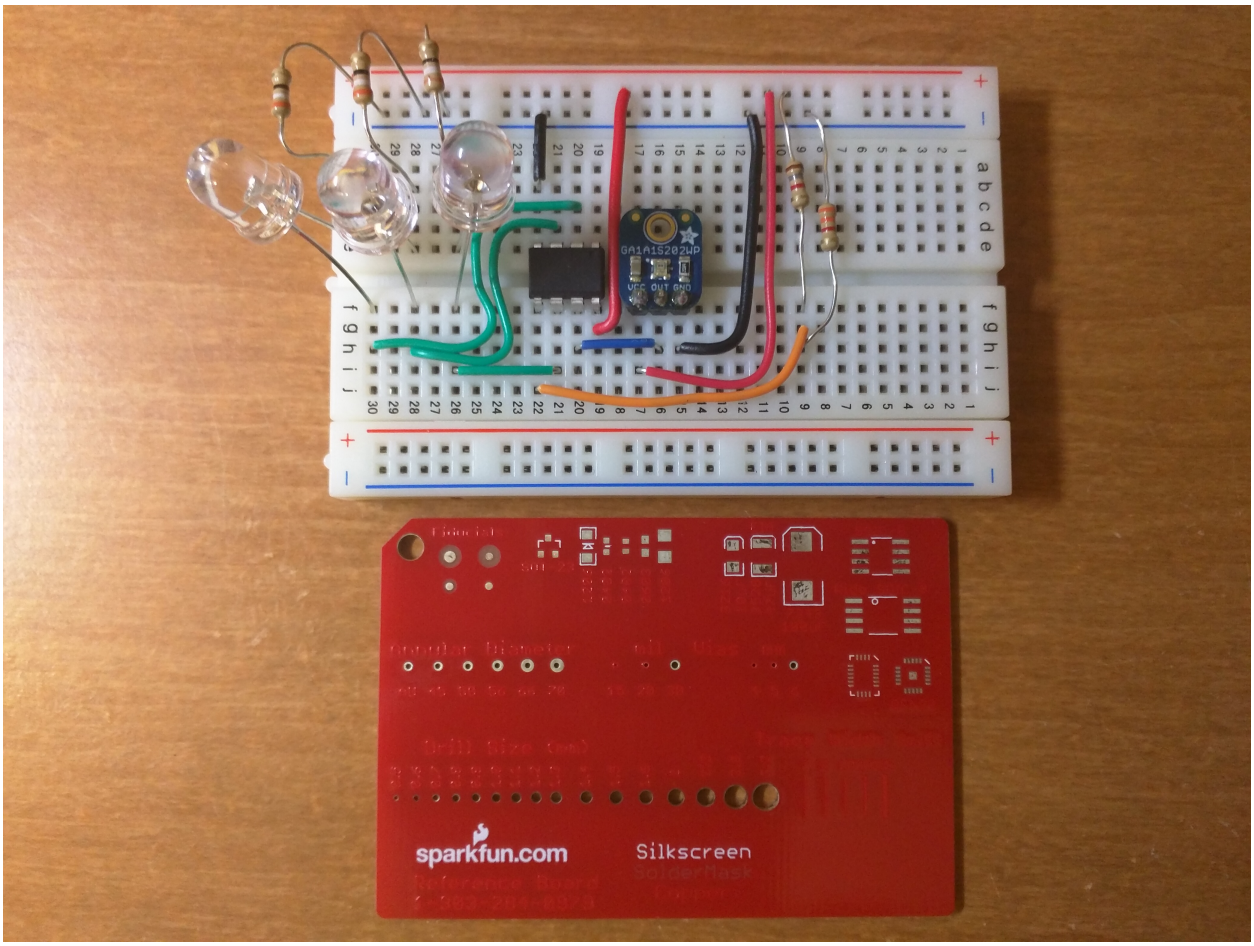


Figure 16: PCB layout planning.

5.1 ATtiny85 Design: 3 LEDs

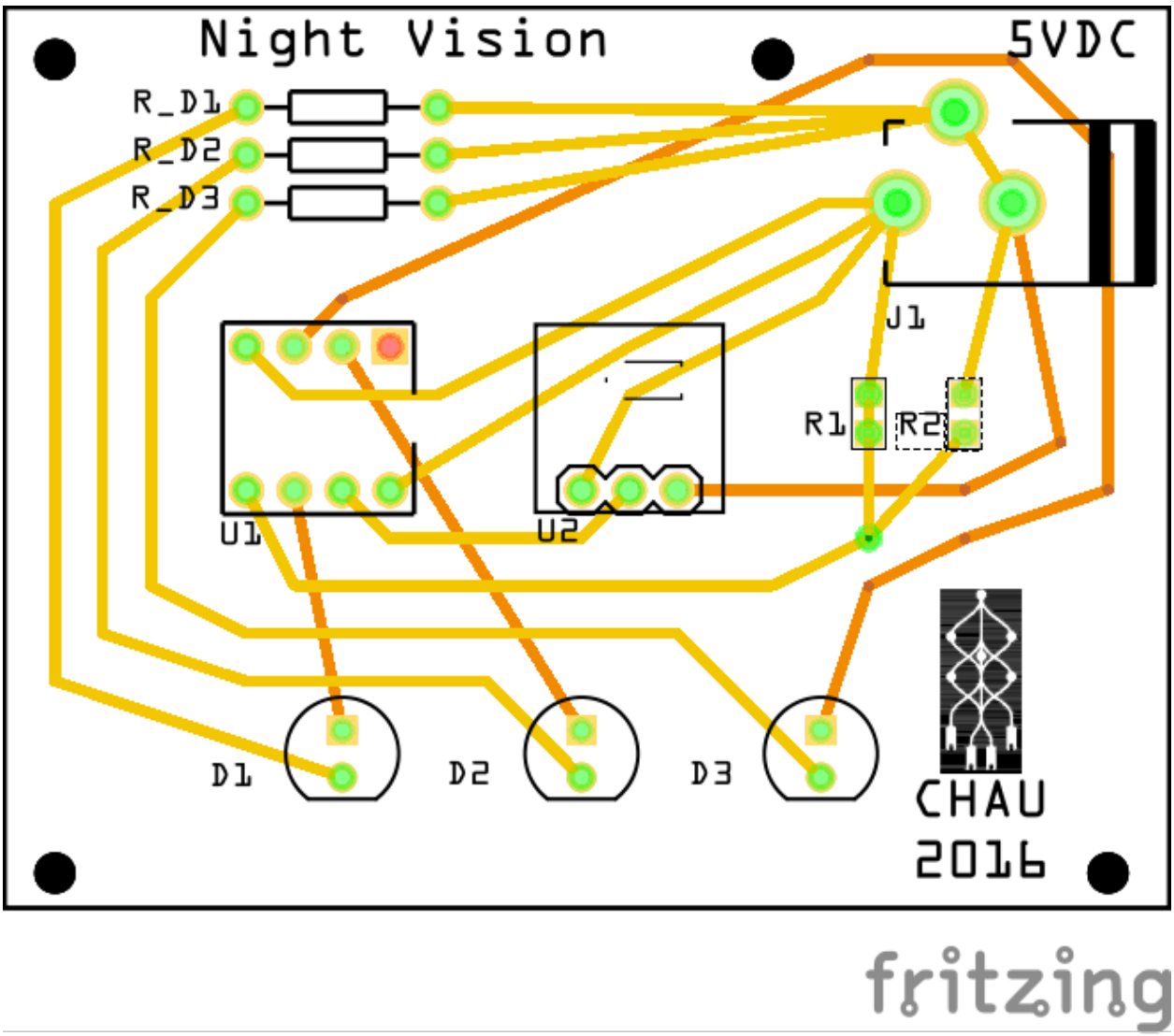


Figure 17: Fritzing PCB layout of the first ATtiny85 design.

5.2 ATtiny85 Design: 6 LEDs

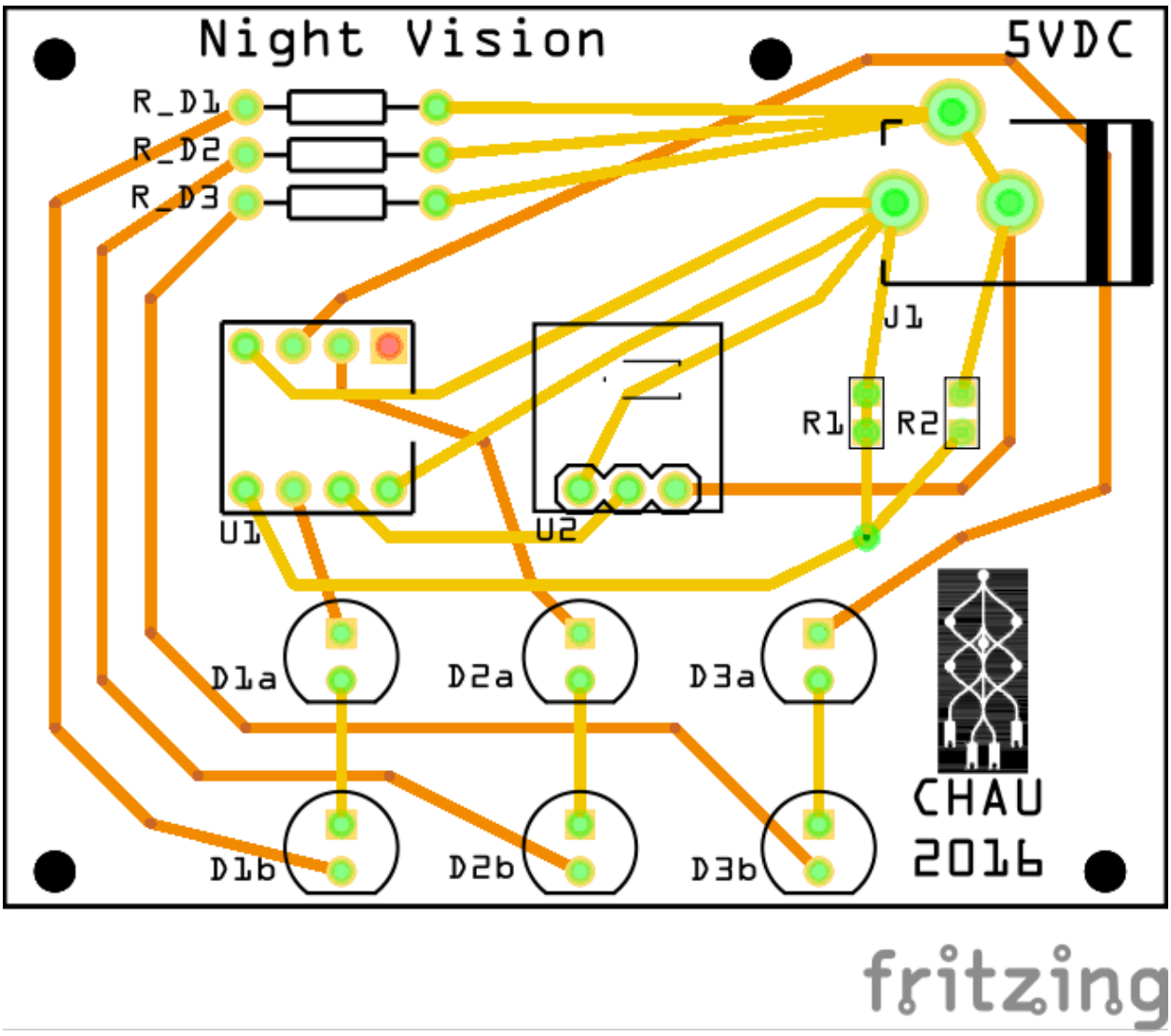


Figure 18: Fritzing PCB layout of the second ATtiny85 design.

References

[1] S. Mallat, “Fourier Kingdom: Properties”, in *A Wavelet Tour of Signal Processing, 2nd ed.*, Chestnut Hill, MA: Academic Press, 1999.

[2] P. Viola, M. Jones, “Robust Real-Time Face Detection”, in *Int. Jour. Computer Vision*, vol. 57.2, pp. 137-154, 2004.

[3] A. Haselhoff, A. Kummert, “A vehicle detection system based on Haar and Triangle features”, Intelligent Vehicles Symposium, 2009 IEEE, Xi’an, pp. 261-266, 2009.

[4] H. Bai, J. Wu, et al, “Motion and haar-like features based vehicle detection”, *Multi-Media Modelling Conference Proceedings, 2006 12th Int.*, Beijing, pp. 4, 2006.

[5] R. Kostia, “Video-based traffic monitoring at day and night time: Vehicle features detection and tracking”, in *Proc. 12th International IEEE Conf.*, pp. 285-290, 2009.