論　文

Development of a Line Following Robot with Human Face Detection System

チェンダー マネット（高知大学教育学部）
北川 晃（高知大学教育学部）

Manith CHENDA and Akira KITAGAWA

Faculty of Education, Kochi University

ABSTRACT

We report a development of a line following robot with human face detection system. This robot can detect a human face while it is tracing a black line drawn on a white field. The human face detection system is connected to the microcontroller of the robot, and we can also control the behavior of the robot through the signal triggered when a face is detected. This robot moves according to the following scenario, like a taxi: (i) The robot cruises along a routine course while searching for a human face. (ii) When the robot detects a face, it stops to pick up the person. (iii) The robot moves again after a moment to take the person to the destination. The present device is supposed to be a teaching material for electronics and programming classes in a high school course, hence it is entirely built up and programmed with easily obtainable components and free libraries, respectively. The present device will be useful in the electronics and programming education for young people.

I. Introduction

We can find a lot of electronic devices here and there nowadays. The modern civilization is based on electronics, and it is necessary for us to understand the essence of that more or less. In this meaning, the education of electronics is very important for young people in their school days. However, electronics is a highly systemized field, and it is difficult to study the field in a short time. As a result, young people tend to lose their interests in electronics. To foster a greater interest in the field of electronics within them, various kinds of teaching materials have been researched and developed.

One of them is the line following robot. This kind of robot detects a line drawn on a field by optical sensors and it traces the line. In the past a number of this type of robots have been studied, and control schemes for them are well established. We can design a line following robot easily with the use of a commercially available controller, for example, Arduino. Here we add another example that contains a trendy feature, that is, machine learning.

Machine learning, which allows computer to learn and improve the performance of a system, used to be a fantasy in the past has been realized and integrated into our daily lives[1]. It evolves from different domains, one of which is control theory. Control theory is a field which optimizes the performance of a system by comparing feedback data with desired output and generating new input to reach that desired output. A good way to understand how it works is to implement it on a real hardware.

The present robot we propose here is capable of detecting a human face, even if it is an initial look, because the robot has a large number of samples as learning data. We control the robot by utilizing this detection system. The present scenario is as follows:

1. The line following robot is cruising along a routine path.
2. The robot searches a human face during its cruising, like a taxi.
3. When the robot finds a human face, it stops to 'pick up the person' as a customer.
4. A while later, the robot again starts moving to take the customer to his/her destination.

We implemented this detection system with the Raspberry Pi (RPi) 3 and its camera module (RPi camera). The RPi 3 is a credit card-sized computer and it is managed by a Linux based Operating System, that is, Raspbian[2]. We also used the Open Source Computer Vision (OpenCV) package, which is an open-source library for image processing and analysis including machine learning. This package is available through the Python language.

Here we suppose that the line following robot with human face detection system is employed as a teaching material in a high school course. All components of this robot are designed with open-source software and components that we can obtain easily. The present robot is related to not only electronics, but also programming, especially for machine learning system. The present teaching material would be appropriate for education of electronic control devices.

This paper is organized as follows. In Sec. II, we show a line following robot that we developed. Here, hardware design, calibration of sensors, and line following algorithm design are included. In Sec. III, we explain a human face detection system. In Sec. IV, we show how the present robot operates on a test field. Section V is devoted to a summary.

II. Design of the present line following robot

We show the outline of the present line following robot (Fig. 1) in this section.

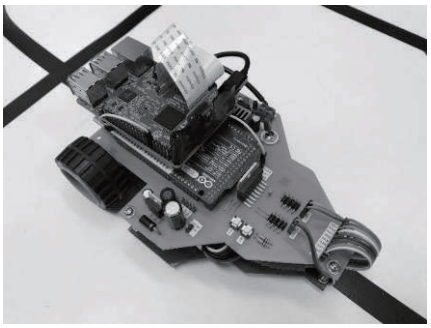


Fig. 1: Prototype of the line following robot.

A. Hardware design

The present line following robot can detect and follow a black line drawn on a white field. The information of the line is obtained as analog signals, which are generated by five optical sensors (TCRT5000). They are the eyes of the robot and are attached at the bottom along a horizontal line as shown in Fig. 2. Note that the most left and right sensors are currently not used. This sensor layout is employed to realize an accurate control of the body. They should be calibrated before every operation, according to a manner explained in the following section. Refer to Appendix 1 for the schematic of the sensors board.

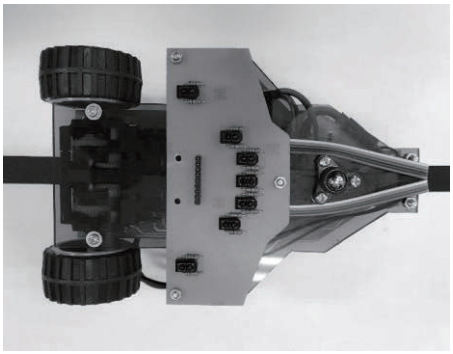


Fig. 2: Bottom view of the robot.

The signal detected is input into a microcontroller. This device controls two motors that are directly connected to left and right wheels through motor drivers (L298N). The two motors are driven under a 3V source, which is regulated from a 12V battery. Here we choose the Arduino Mega 2560 as the microcontroller. This model is characterized by its user-friendly Integrated Development Environment (IDE) and built-in serial-to-USB adapter. These features allow us to conveniently interface the microcontroller through a computer. Refer to Appendix 2 for the schematic of the robot's main board.

B. Sensors calibration

Proper sensor reading values are essential for the control system. Each sensor value is supposed to have the same minimum and maximum value. Moreover, according to the optical sensor's characteristic, in a range of 1 to 1024 of the 10-bit analog-to-digital conversion (ADC), on the white reflective surface, its minimum value should be close to 1, while on the dark black surface its maximum value should be close to 1024. However, due to the imperfection of the hardware, it will never reach such an ideal range, especially the maximum value.

The calibration is carried out through the following equation[3]:

$$x_{cal} = \frac{(x_{actual} - x_{act\,min})(x_{max} - x_{min})}{x_{act\,max} - x_{act\,min}} + x_{min}, \quad (1)$$

where x_{actual} is the actual reading value of the sensor. $x_{act\,min}$ and $x_{act\,max}$ are respectively the actual minimum and maximum reading value of the sensor. x_{min} and x_{max} are respectively the desired minimum and maximum value of the sensor. This equation transforms the actual sensor reading value x_{actual} in a range of $x_{act\,min}$ and $x_{act\,max}$ to x_{cal} in a desired range of x_{min} and x_{max}.

Computing $x_{act\,min}$ and $x_{act\,max}$ can be tricky. These two parameters need to be computed only once during the program execution in two cases within a period of time like 5 seconds or until there is an external interrupt. They are computed by the following conditions:

1. $x_{act\,min} = x_{init}$ if $x_{init} < x_{act\,min}$, (2)
2. $x_{act\,max} = x_{init}$ if $x_{init} > x_{act\,max}$, (3)

where x_{init} is the initial reading value of a sensor.

Let's look at the following two case studies where first an optical sensor is placed on a white reflective surface and then on a dark black surface. Initially, $x_{act\,min}$ and $x_{act\,max}$ are set to 1023 and 0 respectively. These may seem backwards, but it is a trick to compute their actual values.

First, in Case 1, the sensor is on the white reflective surface, and we obtain $x_{init} = 20$. Therefore, from Eqs. (2) and (3):

1. $x_{init} = 20 < x_{act\,min} = 1023$ is true $\Rightarrow x_{act\,min} = 20$, (4)
2. $x_{init} = 20 > x_{act\,max} = 0$ is true $\Rightarrow x_{act\,max} = 20$. (5)

Now $x_{act\,min} = x_{act\,max} = 20$. Then, Case 2 occurs where the same sensor is on the dark black surface, and we obtain $x_{init} = 850$. Hence, from Eqs. (4) and (5):

1. $x_{init} = 850 < x_{act\,min} = 20$ is false $\Rightarrow x_{act\,min} = 20$,
2. $x_{init} = 850 > x_{act\,max} = 20$ is true $\Rightarrow x_{act\,max} = 850$.

After all, we obtain $x_{act\,min} = 20$ and $x_{act\,max} = 850$ for Eq. (1) to calculate the x_{cal} of one sensor. The same method is applied for other sensors.

C. Line following algorithm

The general idea of the line following robot is to keep the robot on a drawn line while it is moving forward. This can be done by individually varying the rotational speed of each motor, so that the robot can turn left or right back to the line when it is about to turn away. Therefore, one needs to set a central point of the robot as a reference to know whether the robot is turning away. This point is supposed to be aligned with the drawn line all the time.

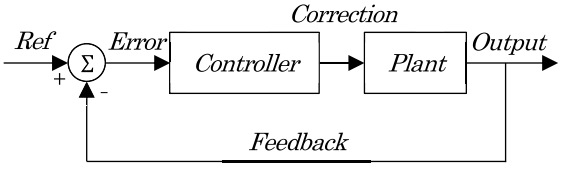


Fig. 3: Close-loop control system block diagram.

Figure 3 illustrates the control system block diagram of the line following robot, where the *Ref* (Reference) is the preset desired central point of the robot, while the *Output* is the actual position of the central point of the robot compared to the line in real time. The *Error* is the difference between the *Ref* and the *Output*, and it is a parameter of the *Controller*. The *Controller* corrects the error by varying the speed of each motor. The *Plant* describes the characteristic of the robot – when both left and right motors have a different speed, it causes the robot to turn, and the outcome of this phenomenon is the position of the robot on the field.

The question is how we could determine the actual position of the central point of the robot compared to the line. The answer is the sensors array. We can imagine oppositely from the reality by assuming that the black line is moving on the array of optical sensors as shown in Fig. 4 below.

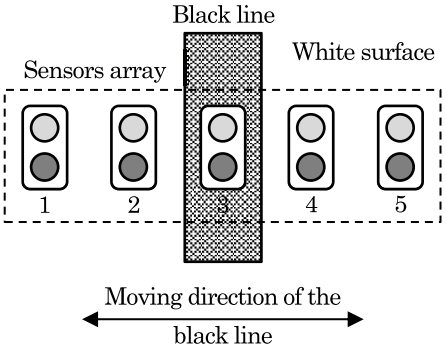


Fig. 4: Optical sensors array versus black line.

With this assumption, we can determine the location of the black line on the sensors array using Centroid Method described by the following equation[4]:

$$C = \frac{\sum_{i=1}^{n} i x_i}{\sum_{i=1}^{n} x_i}, \tag{6}$$

where i is the order of the sensor counting from left to right, and x_i is the reading value of the i^{th} sensor.

The result of this equation is utilized as a feedback for computing the *Error* of the control system. Since the 3rd sensor is in the middle of the sensor array, it is picked as the desired central point of the robot. Therefore, *Ref* equals to 3.

Let's say the black line is now somewhere at the right side of the sensors array. Any sensor that is close to or right on top of the black line will generate a high reading value, while the rest that are on the white reflective surface will generate low reading values. Table 1 provides a set of sensor reading values for a case study.

Table 1: A set of sensor reading values for a case study.

Sensor i	1	2	3	4	5
Value x_i	40	50	800	1020	190

From Eq. (6), we get $C = 3.6$ meaning that the black line currently concentrates between the 3rd and 4th sensor, and the robot is turning away to the left direction. From here, the *Error* is:

$$e(t) = Ref - C = -0.6. \tag{7}$$

To correct the error, the classical Proportional-Integral-Derivative (PID) Controller is applied, and its formula is described by [5]:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)\, d\tau + k_d \frac{de(t)}{dt}.$$

However, due to the fact that the program execution in the microcontroller is sequential, the equation can be simplified as:

$$\begin{aligned} Correction = k_p e(t) + k_i[e(t) + e(t-1)] \\ + k_d[e(t) - e(t-1)]. \end{aligned} \tag{8}$$

Here k_p, k_i and k_d are the coefficients of Proportional, Integral and Derivative Controller respectively. They need to be tuned manually by observing the performance of the robot. $e(t)$ is the current error and $e(t-1)$ is the previous error.

The result of the *Correction* is then utilized for calculating the Pulse Width Modulation (PWM) value to drive each motor. Since the PWM resolution of the Arduino Mega 2560 is 8 bits (0 to 255), PWM outputs for left and right motors are written as follows:

$$\begin{aligned} Left(PWM) = 255 - Correction \in [0, 255], \\ Right(PWM) = 255 + Correction \in [0, 255]. \end{aligned} \tag{9}$$

Note that the PWM value needs to be constrained between 0 and 255. Let's study this case further in number by letting $k_p = 100$, $k_i = 0$ and $k_d = 0$ so that $e(t-1)$ is no longer a concern. Hence, from Eqs. (7) and (8), we get:

$$Correction = -60. \tag{10}$$

Substituting Eq. (10) in (9), we get:

$$\begin{aligned} Left(PWM) = 255, \\ Right(PWM) = 195. \end{aligned}$$

From these results, we can understand that the right motor's rotational speed is now slower than that of the left motor making the robot turn right as an oppose to its previous error where the robot was turning away from the line to the left direction, and thus minimize the error.

III. A detection system of a human face

Here, we describe the outline of a detection system of a human face. This function makes the present robot unique in comparison with previous ones. This system is additionally equipped on the line following robot explained in the previous section, and it is connected to the Arduino device to control the action of this robot.

This human face detection system is implemented by the RPi 3 and its camera module.

A. OpenCV library

Open Source Computer Vision (OpenCV) is an open-source library containing several hundred algorithms for computer vision applications[6]. Some modules of the library being used in the present work are as follows.

- Core functionality: defines basic data structure and functions used by all other modules.
- Image processing: converts image color, draws rectangles around detected objects, and so on.
- Object detection: compares input image to predefined classes such as faces, eyes, and so on.
- High-level GUI: displays images, and interacts the program with hardware (in this case, a keyboard) for the program termination.
- Video I/O: captures and saves a video (used only during the test on a PC).

This library is utilized entirely in the Python working environment.

The video I/O module can only be used for USB camera or webcam, while the RPi camera module communicates with the RPi via a different communication protocol. Hence, to access the camera module effectively, one needs the Picamera library.

B. Detecting a human face

The face detection is carried out based on a fast and effective Haar Feature-based Cascade Classifier initially proposed by Viola and Jones[7]. Firstly, a classifier or a detector is trained with up to thousands of positive and negative examples. The term positive example refers to an image containing the object of interest – in this case a face, while the negative example is the image without that object of interest. This classifier is then applied to an input image for searching the object of interest.

Implementing the face detection in Python with OpenCV is quite straightforward. First, a frontal face classifier has to be loaded. In our case, we use an available pre-trained frontal face classifier. Next, we continuously capture an image. Each image is converted to grayscale. This is due the fact that the classifier is also trained by grayscale examples (Haar Feature is applicable only for grayscale images). Finally, the classifier is applied to search for faces in the grayscale input image. When faces are detected, it returns the size and coordinate of the rectangle surrounding each face in the input image. This information is usually utilized for drawing a rectangle around the detected face while displaying the detection result, but in our case, we also used it to stop the robot when the size of the rectangle reaches a certain value.

IV. Operation of the line following robot with human face detection system

This section reports the operation of the current development. We prepared a test course and a photo panel of a human face as shown in Fig. 5. The test course consists of a white field (size: 90cm×90cm) and a black line drawn on it.

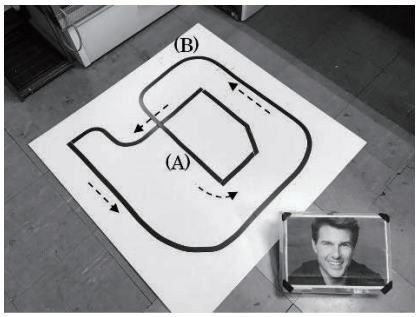
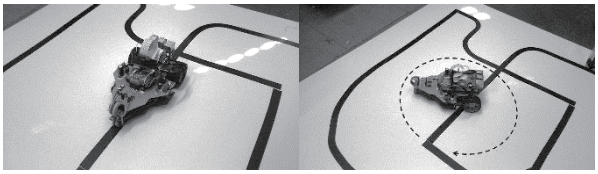


Fig. 5: Test course with arrows representing forward direction of the robot and a photo panel of a human face.

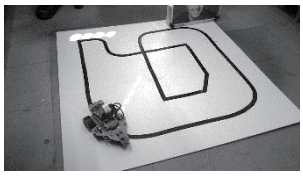
As the first step, we put the line following robot at the start point (A) (Fig. 6 (a)). The robot automatically rotates clockwise at the point and calibrates its own sensors according to procedure explained in Subsec. II-B (Fig. 6 (b)). After the calibration, the robot starts moving along the black line and keeps cruising for a while (Fig. 6 (c)). We then put the panel of human face at an arbitrary point, for example the point (B). The robot detects the face according to the algorithm explained in Subsec. III-B. When it gets close to the panel at about 20 cm long distance, it stops in order to 'pick up a person as a customer' (Fig. 7 (a)). A while later, the robot continues its journey along the black line (Fig. 7 (b)).

The face panel is used as a signal to stop the robot. The robot always keeps searching for a human face via the camera module during its cruising. The present robot can

detect a human face while it is moving, and this feature is characterizing the present system. It should be noted that the human face detection procedure is treated before stopping. This feature allows the robot to move smoothly.



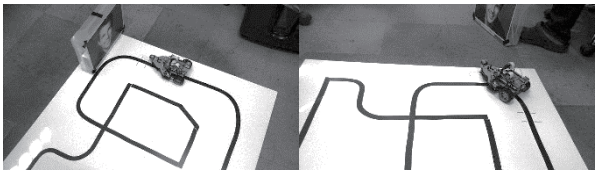
(a) (b)



(c)

Fig. 6: (a) The robot is placed at point (A). (b) It then rotates itself for calibrating sensors. (c) The robot is cruising on the field.

We suppose the present line following robot as a teaching material in a class for electronics and programming in a high school course. The present device consists of two of main functions; line following process and human face detecting process. To build up and set up each assembly, we need works for electronics and programming, which are suitable for training. The present robot is built up with easily obtainable components and programmed by using free libraries of Python programming language. This point also makes the present robot accessible. We believe that the present robot assists young people to study electronics and programming fields effectively.



(a) (b)

Fig. 7: (a) The robot stops when it gets close to the detected face. (b) A while after the face panel is removed, the robot continues its journey.

V. Summary

We developed a line following robot on which a human face detection system is equipped. This robot detects a black line drawn on a white field through optical sensors and the information obtained is sent to a microcontroller to drive motors that are connected to left and right wheels directly. A calibration procedure is necessary for optical sensors before every operation.

The microcontroller is also connected to the human face detection system, which is implemented by the Raspberry Pi 3 and its camera module. This system is programmed by Python language with OpenCV that is a free library. The present robot can detect a human face while it is moving and this information is used as a trigger for stopping the robot.

The present robot is supposed to be a teaching material for electronics and programming in a high school course. This robot can be built up and programmed by easily obtainable components and free libraries only, respectively.

Acknowledgment

The authors would like to thank Prof. H. Doho for his generous gift of electronic parts. This work was supported by JSPS KAKENHI (Grants No. 17H01981).

References

[1] E. Alpaydin, Machine Learning: The New AI, The MIT Press, 2016.

[2] S. Monk, Programming the Raspberry Pi: Getting Started with Python, McGraw-Hill Education, 2016.

[3] B. Earl, "Calibrating Sensors," [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/calibrating-sensors.pdf.

[4] S. Supriadi, A. Rizal , D. S. B. Utomo and A. Wijiansyah, "Line Following Robot Optimization based Fuzzy Logic Controller Using Membership Function Tuning," *International Journal of Engineering & Technology,* vol. 7, no. 2, pp. 112-116, 2018.

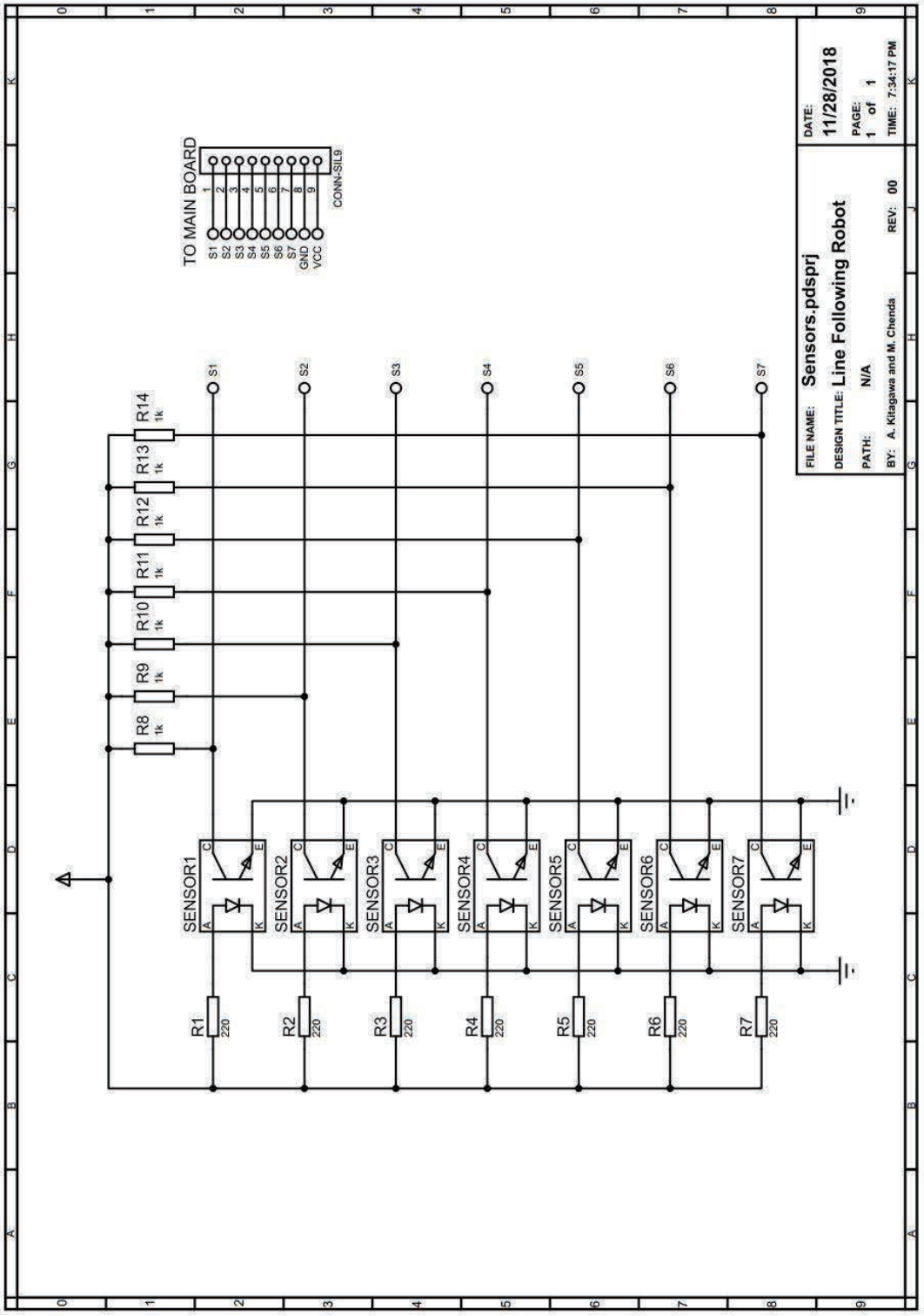
[5] K. J. Astrom and R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, Princeton University Press, 2009.

[6] Intel, "OpenCV," 2 November 2018. [Online]. Available: https://docs.opencv.org/3.4/d1/dfb/intro.html.

[7] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Computer Vision and Parttern Recognition Conference*, 2001.

Appendix 1

The following is the schematic of the line following robot's sensors board.



Appendix 2

The following is the schematic of the line following robot's main board.