ECE 445 Senior Design

Autobrake Bicycle

Individual Report

By

Team 50

Taiqing Ling (taiqing2)

TA: Xihang Wu

1. Introduction

1.1. Team Project Overview

The project of Autobrake bicycle aims to develop a riding assistance for bicycles that could intelligently detect STOP signs and push the bicycle brake to protect cyclists' lives. To be more specific, this project will have a detection subsystem installed on the bicycle, which is able to run a computer vision algorithm for STOP sign detections, and to measure the speed of the bicycle. This detection subsystem will then be wirelessly connected to the warning subsystem on the helmet, which is responsible for generating warning signals to notify cyclists about the existence of a STOP sign. These two subsystems are wirelessly connected through a pair of XBEE modules. Finally, we have a brake subsystem on the bicycle, which should be triggered if the STOP sign is too close while the speed still remains high.

2.2. Individual Responsibilities and Role

During the last few weeks, I was actively working on the detection subsystem, the wireless connection, as well as the brake subsystem. I have personally achieved great accomplishment during my work. The wireless connection was stable and efficient, with imperceptible laggings in byte communications. The brake subsystem requires a voltage up to 12V to trigger, and I have tested the control logic for that by using a motor as a simulated load, which was proved to be successful. The remaining work on this part is to simply install the mechanical brake onto our bicycle and to do several road tests to determine an appropriate voltage to trigger it. The detection system, however, as the most complicated subsystem in this project, still needs some further improvement. So far, I have implemented a python program for computer visions and serial communications with the microcontroller. This program was proved to be accurate and fast on the computer. It can detect an image of 640*480 pixels within 0.05 ~ 0.24 seconds. No false alarm has been discovered so far. Nonetheless, if I use a Raspberry Pi instead to run this program, it becomes significantly slower. I will have to do further improvement on my algorithm or use a Pi with larger memory instead to run the codes. For the speed detection, I use a hall-effect sensor and a piece of magnet to measure the RPM of a wheel. It had a good performance on my test circuit and our next challenge is to find a place on the bicycle to install it while the magnet and the sensor are closed enough to maintain the performance.

2. Individual Design Work

2.1 Design considerations

There are two major changes in our design from the one we stated in our design document. The first one occurs at the detection subsystem. My old plan was to directly upload my detection algorithm to Atmega328, which was actually impossible for two reasons. First of all, the Template Match algorithm is written in Python to be concise, while Atmega only supports C style programming, which could make the code incredibly long and sophisticated. Second of all, the Atmega only has a memory size of 32KB, while a 640*480-pixel gray-scaled image already occupies 300KB, which makes it impossible to be handled on the Atmega. My modified plan is to use a Raspberry Pi instead as a microcomputer to run the algorithm and then use a USB-Serial adapter to inform Atmega whether or not a STOP sign is detected. By doing so, I also got rid of the OV7670 camera module, which is also too slow for video capture. I used a USB-camera instead that was directly plugged into the Raspberry Pi to provide video images. As I illustrated in the sec 2.2, although apparently better than my old plan, the Raspberry Pi is still not performing as well as the computer. Further improvements need to be done on the detection efficiency. The second change occurs on the braking method. We planned to replace the servo in our design document with an electrical magnet, which offers greater power and is easy to install.

2.2 Block Diagram

Fig.1 shows the updated block diagram of our new design. The Raspberry Pi will be powered by a standard battery, which is provided by the Raspberry Pi official to protect the CPU. The camera is a USB device and thus directedly gets power from the Pi. The MOSFET acts as a digital switch for the electrical magnet and is controlled by a digital signal from Atmega. The buzzer is used to generate warning sounds and it does not need extra power.

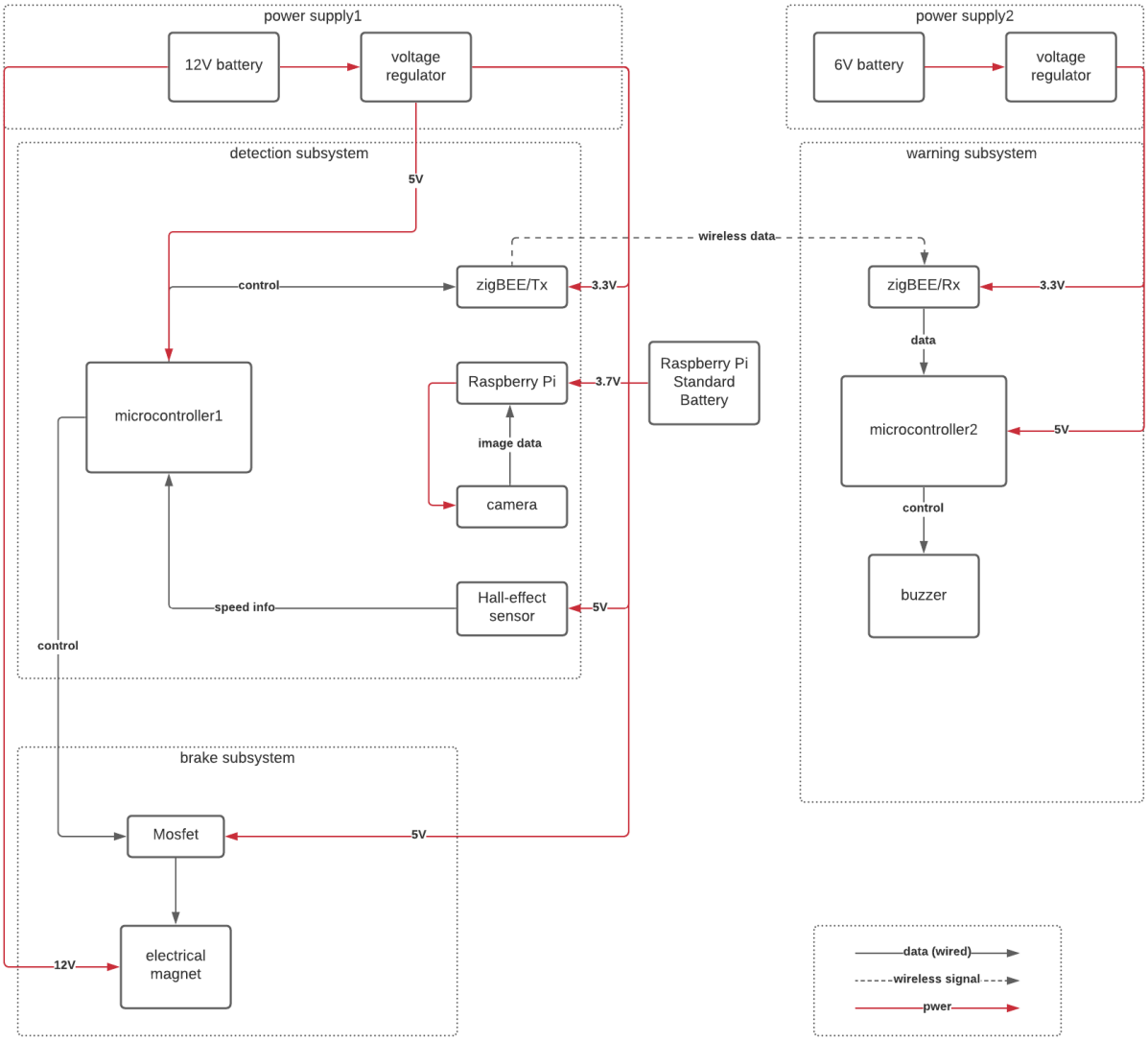
power supply1
12V battery
voltage regulator
power supply2
6V battery
voltage regulator
detection subsystem
warning subsystem
5V
wireless data
control
zigBEE/Tx
3.3V
zigBEE/Rx
3.3V
data
microcontroller1
Raspberry Pi
3.7V
Raspberry Pi Standard Battery
microcontroller2
5V
image data
camera
control
buzzer
speed info
Hall-effect sensor
5V
control
brake subsystem
Mosfet
5V
electrical magnet
12V
data (wired)
wireless signal
pwer

Fig.1

2.3 Template Match and Testing

Template Match is an algorithm I used for computer vision tasks. It serves to find a matching of template from a larger image.

Fig.2

Fig.2 shows the templates I used for STOP sign detection. As searching within a rectangular shape gives the greatest efficiency, I manually cut the "STOP" characters out form the image and use them as my templates. If instead, I use a square including the entire octagon, as shown by Fig.3, it will leave four blank triangles (blue) at the corners as noises, which lead to severe false alarms.

Here are some pseudo codes for using this algorithm:

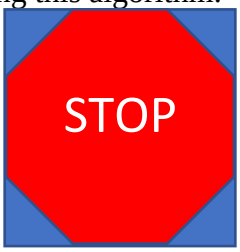

Fig.3

1. Create a list of template images called 'tmplts' and gray scale each template.
2. Create an independent thread to capture images from the camera and always store the latest image into 'frame', which is a shared variable; Gray scale the frame;
3. Initialize Found to False;
4. Fetch the latest frame;
5. For each img in tmplts:

   Run opencv.match_template to get a correlation matrix for img on frame.

   If correlation.max() > 0.58:

   Found=True

   Break

6. if Found==True:

   write 1 to USB port and draw the detected image to verify if it is false alarm.
7. Loop back to step 4;

The correlation threshold 0.58 is set after several tests and was guaranteed to minimize both false positives and false negatives. The independent thread for camera was designed to decrease the time cost of image fetching to lower than 1ms. Furthermore, I used gray scales on both templates and frames because processing only on illumination was proved to have exactly the same results as processing on all three channels of RGB. By gray scaling, the detection speed is tripled. Fig.4 shows a sample test result. The white box on the STOP sign marks the location of detection.

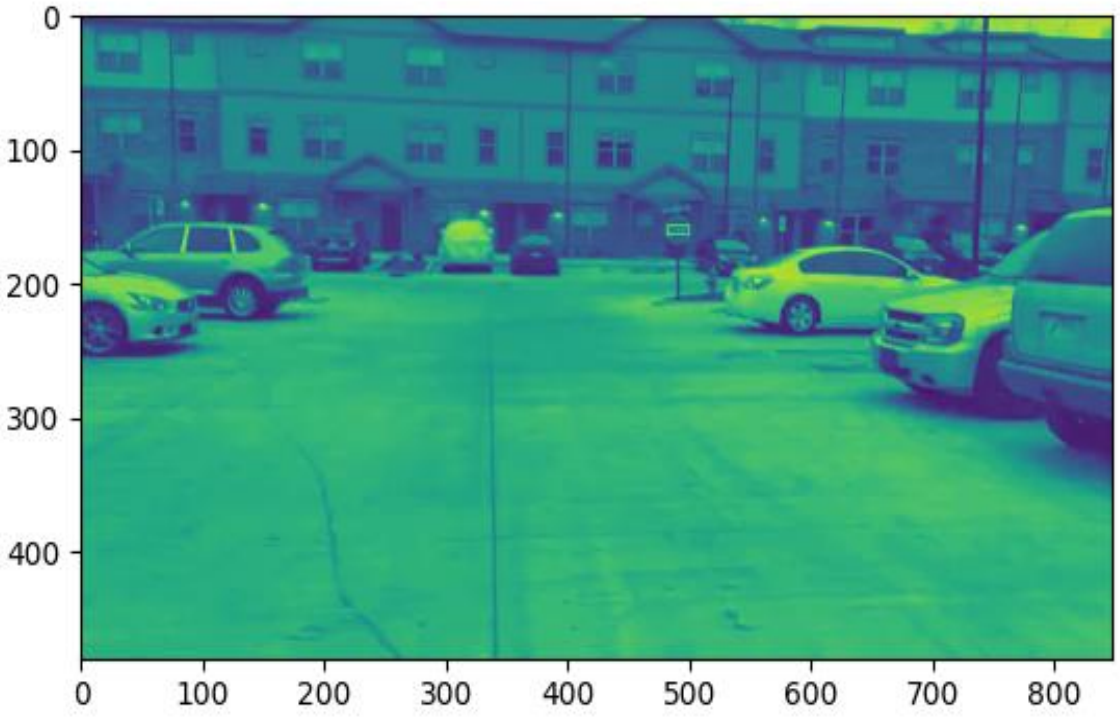

Fig.4

2.4 Wireless Connection Test

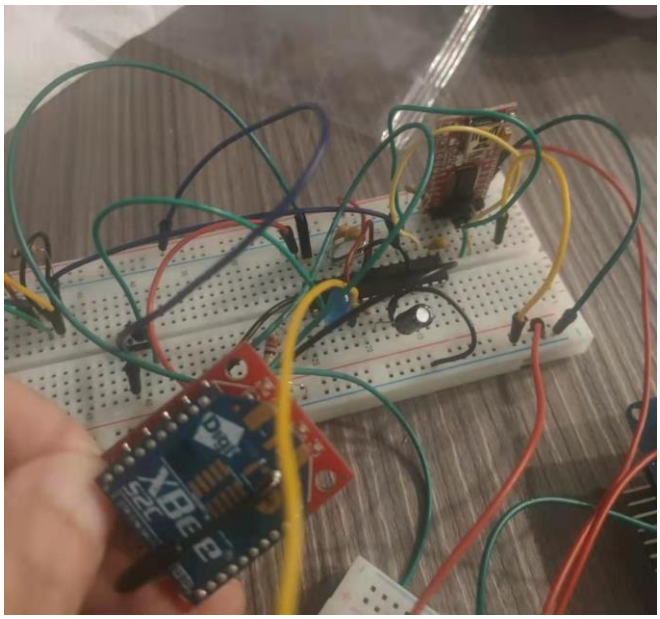

Fig.5 Testing circuit for Atmega and XBEE

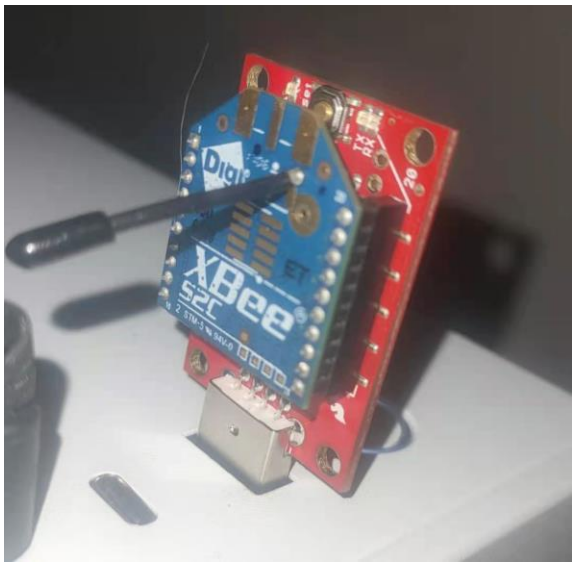

Fig.6 XBEE-to-USB adapter

I built the circuit in Fig.5 on the breadboard to test the wireless connection between XBEEs. The XBEE on the breadboard is a coordinator, which receives the serial byte '1' from the computer and sends a byte 'F' to the end device in Fig.6. The computer console then prints 'F' to indicate that the connection was successfully built.

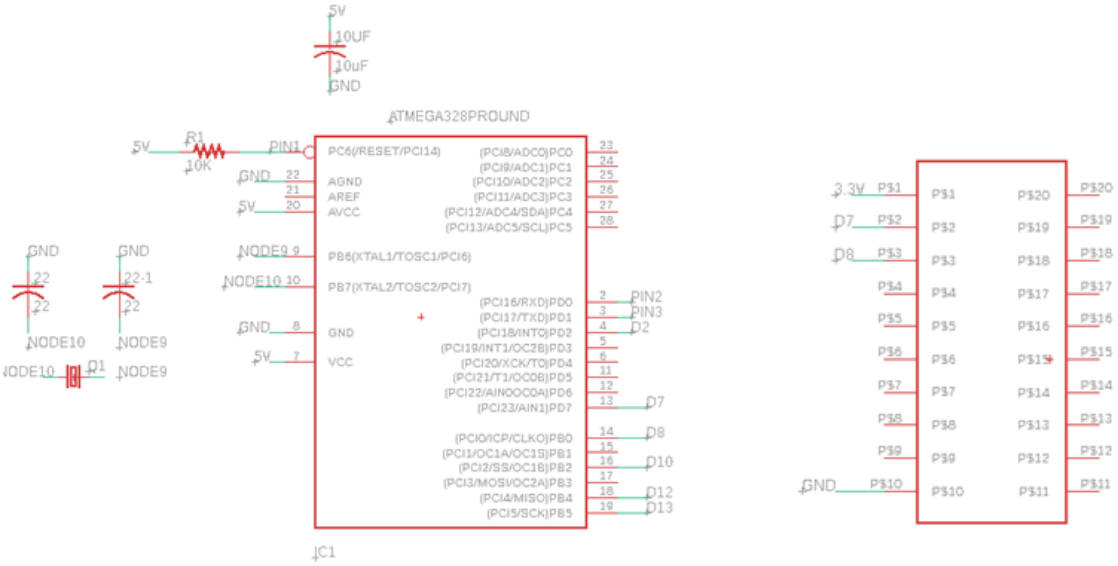

Fig.7 Schematics of Atmega and XBEE coordinator

2.5 PCB design

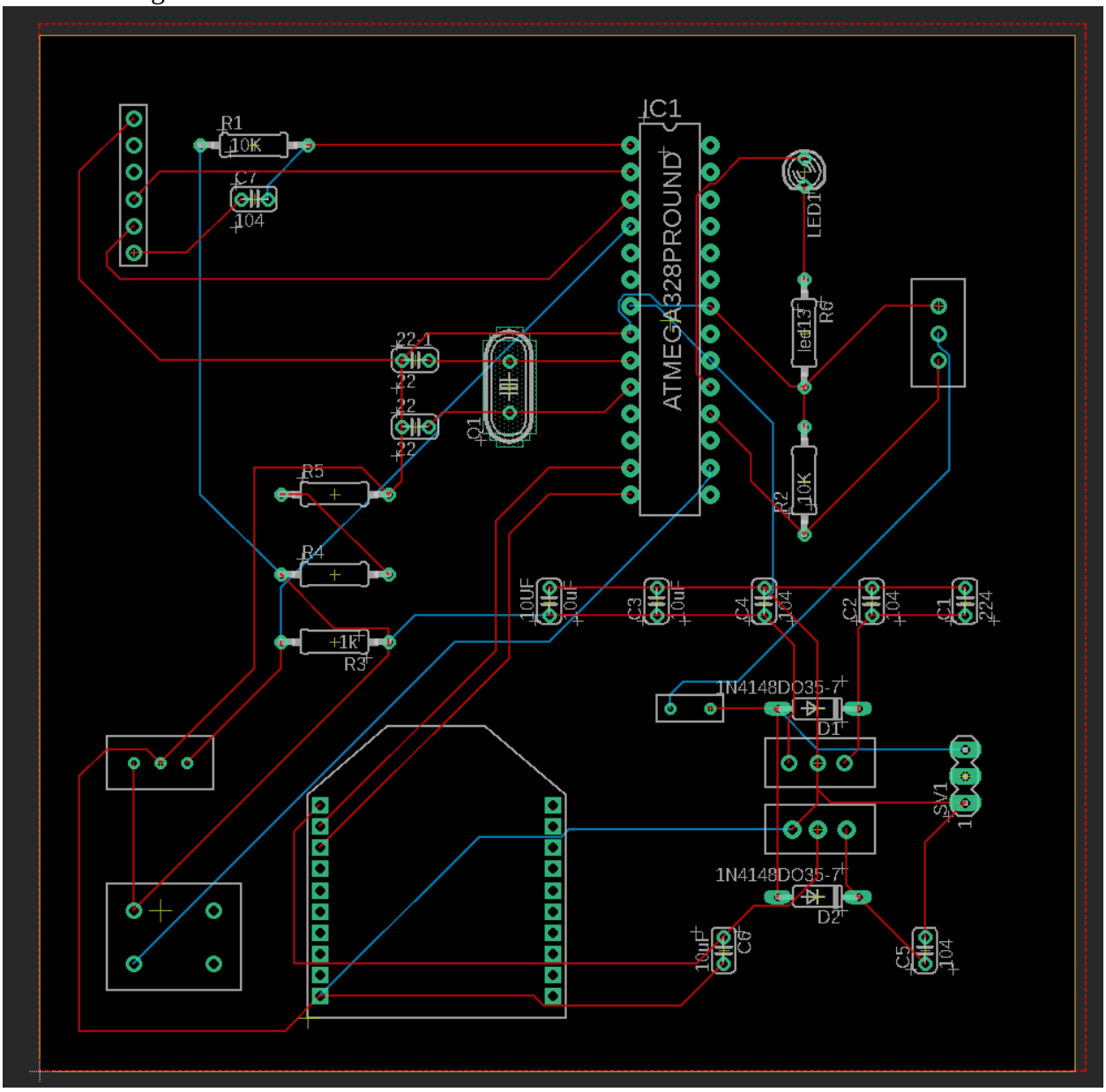

Fig.8 PCB footprint

After doing enough breadboard tests, I finalized the entire design of our circuit. With the help of my teammate, we finished the PCB design as shown in Fig.8.

3. Conclusion

I was playing an active role in the process of both designing and testing. My individual contribution so far for the entire project goes beyond 70% and I have been ahead of schedule. As long as the finalized PCB design work I expected, I am confident that we can succeed on this senior design. For the remaining work, I will focus on improving the detection performance on Raspberry Pi. A rough plan for this improvement is to narrow down the scope of image reasonably. The largest template I used has a size of 50*20 pixels and was seen by the camera at a distance within 5 meters. This implies that the camera has an oversized vision. Some parts of the image may be meaningless to put under detection. (e.x. STOP sign should never show up on the left). By doing such improvement I could probably speed up the detection on Raspberry Pi.

4. References

"Arduino Uno to ATmega328 - Shrinking your Arduino Projects," *YouTube*, 29-Dec-2018. [Online]. Available: https://www.youtube.com/watch?v=Sww1mek5rHU&t=1910s. [Accessed: 04-Apr-2021].

"Template matching." [Online]. Available: https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html. [Accessed: 04-Apr-2021].