

**The Chinese University of Hong Kong  
Department of Computer Science  
and Engineering**

**CENG3430 Rapid Prototyping of Digital System**

**Final Project - FPGA-based Computer Vision  
Accelerator for Train Active Safety System**

**Report**

**Project Demo Deadline: 16 May 2025 23:59**

**1155193237 - Yu Ching Hei (chyu2@cse.cuhk.edu.hk)**

**1155XXXXXX - (1155XXXXXX@link.cuhk.edu.hk)**

**Source code and project history is available on GitHub:**  
[https://github.com/Jellyfish227/FPGA\\_Accelerated\\_Computer\\_Vision\\_on\\_  
Train\\_Active\\_Safety\\_System.git](https://github.com/Jellyfish227/FPGA_Accelerated_Computer_Vision_on_Train_Active_Safety_System.git)

*Under the kind support and guidance of*  
**Prof. Ming-Chang YANG**

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Division of Work</b>	<b>2</b>
<b>3</b>	<b>Design</b>	<b>2</b>
3.1	Transmission of Data . . . . .	2
3.1.1	v1 of data sending formatting(float string) . . . . .	3
3.1.2	v2 of data sending formatting(int string) . . . . .	3
<b>4</b>	<b>Conclusion</b>	<b>3</b>

## 1 Introduction

The CENG2400 final project involved implementing a gyroscope-controlled laser turret that uses the MPU6050 to direct the laser. Our goal was to create a responsive turret for maximizing hits.

The workflow consists of two phases:

1. **Data Pre-processing:** We collect and process gyro and accelerometer data from the MPU6050, then transmit the calculated angles via UART.
2. **Data AI-Inferencing:** The angles are received and converted into PWM signals to control the servo motors.
3. **Data Post-processing:** The PWM signals are received and converted into the corresponding angles to control the servo motors.

This report outlines our challenges faced and insights gained during implementation, aiming to deliver a functional motion-controlled laser turret.

## 2 Division of Work

Task	Person In Charge
Project Proposal	
Data Pre-processing	Yu Ching Hei,
Data AI-Inferencing	Yu Ching Hei
Data Post-processing	
Report	Yu Ching Hei

## 3 Design

### 3.1 Transmission of Data

### 3.1.1 v1 of data sending formatting(float string)

```
1 /* Commit hash: c6e8ed6, main_mpu.c */
2 void sendData(float yawAngle, float pitchAngle) {
3     char data[22];
4     // format the data as a string
5     sprintf(data, "%.10f,%.10f", yawAngle, pitchAngle);
6     char* chp = data;
7     while (*chp)
8         UARTCharPut(UART5_BASE, *chp++);
9 }
```

In this initial implementation, we formatted the yaw and pitch angles into a string using `sprintf`. While this allowed us to send the angles with high precision (10 decimal places), the long string to be sent over UART wirelessly increases the risk of data loss. And turns out during the testing, we found that the data received was very unstable, we have presumed that the problem is due to the data being sent is too long, which means there are more characters that can potentially be lost, and just one character lost can cause the whole data to be corrupted.

### 3.1.2 v2 of data sending formatting(int string)

```
1 /* Commit hash: af09000, main_mpu.c */
2 void sendData(float yawAngle, float pitchAngle) {
3     char data[7];
4     sprintf(data, "%03d%03d\0", (int)yawAngle, (int)pitchAngle);
5     /* same send looping as before */
6     UARTCharPut(UART5_BASE, '\n'); // '\n' to mark end of transmission
7 }
```

In this update, we have made 2 important changes:

1. We have changed the data string to be a string of 7 characters, which is a lot shorter than the previous 22 characters.
2. We have added a newline character to indicate the end of the transmission, which is a simple way to check the data integrity.

We decided to sacrifice the precision of the data to just send the integral part of the angle, in order to reduce the risk of data corruption due to data loss during transmission. And in the previous implementation, we found that we have no way to check the data integrity, so we have to add a newline character to indicate the end of the transmission. However, the problem was not fixed, in the slave end, we found that the data received was still very unstable, we noticed that sometimes the data received would swap their positions(i.e. the yaw angle would be the pitch angle and vice versa), or sometimes some digits that should be in the pitch angle would appears in the array of yaw angle characters received. Therefore, we started to investigate in the slave end.

## 4 Conclusion

To sum up, we have successfully implemented a responsive laser turret that can capture the motion of hand and reflect it to the laser turret wirelessly. Before working on this project, we have expected that the most challenging part would

be the motion capturing part. But surprisingly, we found that the most challenging part was actually the data transmission part, which we have to ensure the data integrity and stability. Through this iterative process of design, implementation, and debugging, we have gained valuable insights into embedded systems development. These experiences and lessons learned will prove invaluable for future embedded systems projects encountered.