Project 2

MECHENG 706 — Group 18

Section #	Section Name	Person
1	Hardware Configuration	Peter Mitchell
2	Software-Hardware Integration	Hamish Giles
3	Obstacle Avoidance	Harvey Merton
4	Fire Sensing and Extinguishing	Vincent Wong

Table of Contents

Summary and Introduction

As the field of robotics evolves, the number of potential applications continues to expand. This continued push for robotics allows for improvement to certain jobs or services that can often be hazardous, where human interaction needs to be minimised. An example of this is firefighting. Firefighting is often very dangerous and the ability to use robots to extinguish fires will greatly reduce the risk to human lives.

MECHENG 706 Project 2 focuses on the application of fully autonomous robotics in a fire extinguishing robot. This robot has to be able to autonomously locate 4 obstacles in its environment and navigate around them without collision in an attempt to extinguish 2 fires (small incandescent light bulbs with resistors on top). Once the fires are located, the robot needs to position itself 20 cm away (centre of robot to centre of obstacle) from the fire prior to extinguishing it. Both fires must be extinguished within 60 seconds.

To allow the development of a robot with the above behaviour, the project is split into various tasks. These tasks are hardware configuration; software-hardware integration and higher-level behaviour control; obstacle avoidance using fuzzy logic; and fire locating and extinguishing. Each task is covered in detail in a separate section of the following report. During the demonstration, the design presented successfully extinguished both fires within 46 seconds while avoiding collision with all obstacles on the track.

1 Hardware Configuration

For the "fire" fighting robot to detect its environment in terms of object detection and also light detection, a range of sensors (light intensity and range sensors) are used. For the movement and actuation of the fan and its position on the robot are made possible by an array of servo motors and controllers. The full list of sensors, actuators and electronic components can be found in Table I below.

Table I: Bill of Materials and Arduino Pin Allocation

Component	Quanti ty	Purpose	Pin Allocation
Vex 393 Motors	x4	Servo Motors to Move Robot	46(LF), 47(LR), 50(RR), 51(RF)
Turret Servo	x1	Rotate Fan Assembly	41
50mm Fan	x1	Fan for Extinguishing "Fire"	-
FQP30N06	x1	Fan MOSFET for On/Off Toggling	9
2Y0A41SK	x2	Medium Range IR Range Sensors	A8(R), A10(L)
2Y0A21	x2	Long Range IR Range Sensors	A9,A11
HC-SR04	x1	Ultrasonic Range Finder	48(Trig), 49(Echo)
ADXRS642	x1	Single Axis Gyroscope for Rotation	A3
SFH 300 FA	x3	"Fire" Detection Phototransistors	A4(L), A5(C), A6(R)

1.1 Actuators

The base of the robot consists of 4 Vex 393 Motors connected to mecanum wheels. This allows for the robot to have linear movement in the x and y directions while still allowing it to rotate. These motors are connected to motor controllers that are connected to analog pins of a motor shield connected to an Arduino Mega 2560. The motor shield allows for extra power to be utilised by the servo motors as the analog pins of the arduino are not rated for high current loads.

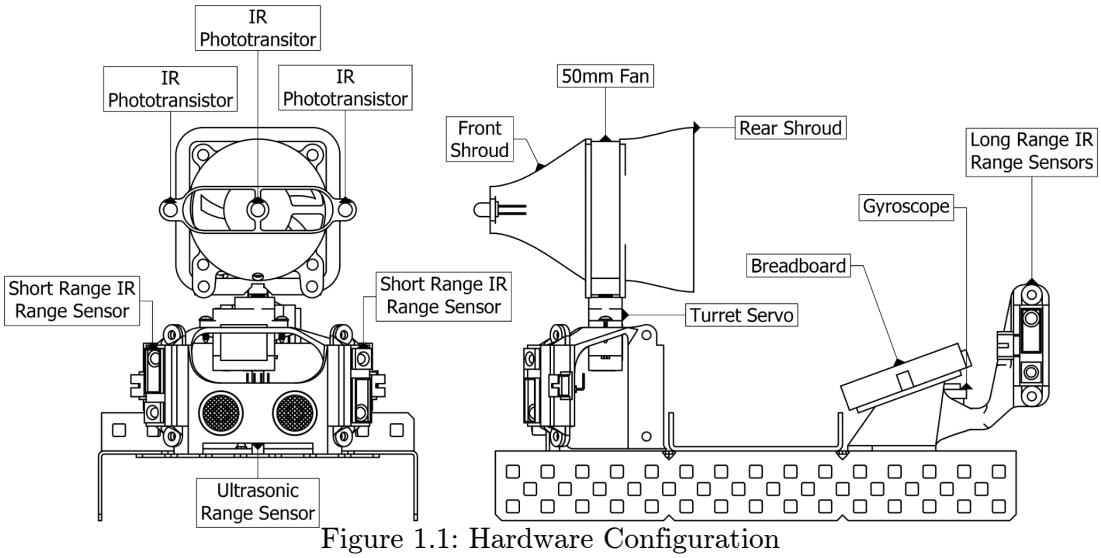

Figure 1.1: Hardware Configuration

The "fire" extinguishing and light sensing system are all integrated into a fan shroud design as seen in Figure 1.1. A micro servo is used for the rotation of this fan assembly that carries the fan and the phototransistors for light detection. Having the fan on a servo allows for precise alignment of the fan to the "fire" to ensure that as much air as possible is getting to the "fire" irrespective of the rotational position of the robot.

1.2 Sensors

The front medium range IR sensors are placed at a 55fl angle from the centre plane of the robot as seen in Figure 1.1. This allows for tracking distance to the front corners of the robot's wheels. The long-range IR sensors are mounted in the back centre of the robot. This allows the sensors to pick up when the back wheels of the robot are about to hit any obstacles. The ultrasonic is used for positioning in the forward direction, as the two medium IR sensors are unable to pick up range to objects from the front of the robot due to the angle that they were placed on. Due to the minimum distance of each of the range sensors (InfraReds and Ultrasonic), each of the sensors were placed in body the same distance as their minimum distance effectively making the minimum distance of the sensor zero centimetres from the bounds of the robot. The gyroscope was not needed for this project as the robot completed the required tasks without any issues omitting the usage of the gyroscope.

This sensor arrangement allows the detection of any objects on the robot's four corners (over the wheels) and also full coverage of the front of the robot. However, with this design there are some inherent "dead zones" that are present where the robot is unable to detect any obstacles. These are on the sides of the robot (in the middle) and on the rear of the robot. This is however not an issue as the robot should only strafe, drive forward and rotate on the spot. This makes it nearly impossible for the robot to hit any obstacles (excluding a couple edge case scenarios). To fix the issue of the side blind spot, a feature map of recently seen obstacles can be made to estimate obstacle location in the vicinity when the robot tries to strafe in a particular direction. This was not implemented as it was found that due to the lack of obstacles that needed to be avoided, this case will almost never happen.

1.3 Circuits

From project specifications, the fan required for the "fire" extinguishing system only needs to be turned on (or spinning) when the "fire" is 20 centimetres from the centre of the robot. Due to this a MOSFET was used to switch on and off the fan with digital logic from the Arduino microcontroller. The MOSFET allows for control of the switching on of the fan with a digital high signal (˜+5v) from the Arduino and switching off of the fan with a digital low (˜0v) signal. It was only of concern to either have the fan at full speed or completely off, meaning digital PWM or analog voltage control to control the speed of the fan was not necessary. The circuit configuration for the fan can be seen in Figure 1.2.

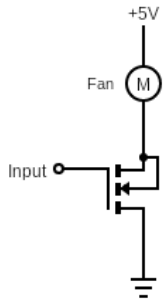

Figure 1.2: Fan Circuit

For the "fire" extinguishing system to detect any "fires", it needs a way to sense them. There are multiple ways to do this, from heat detection to finding the light intensity of the "fire". InfraRed Phototransistors were used to collect any light intensity data from the robots environment. These phototransistors when paired with a resistor can be used as a voltage divider if configured as seen in Figure 1.3. This particular configuration allows for the phototransistors to work in an active high state, meaning that as light intensity increases the voltage at the output also increases. For the arduino to read these light intensity (or voltage) values each phototransistor circuit output is connected to an analog input pin on the arduino. This allows the arduino to process the voltage fluctuations from an analog voltage to a 10-bit reading ranging from 0-1024 (no light - large light intensity).

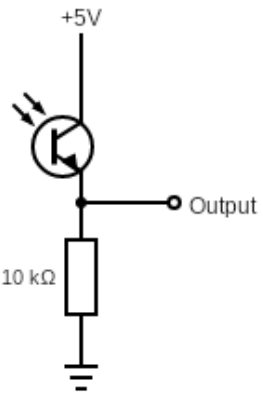

Figure 1.3: Phototransistor Circuit

The other important aspect of the phototransistor circuitry is the resistor paired with the phototransistor. The lower the resistance of this resistor, the less the output voltage will change with respect to the light intensity. With this knowledge in mind, a range of resistors were tested to determine what the optimal resistance is for the requirements of this "fire" fighting robot. Resistors from 100 ohms to 100,000 ohms were tested and ultimately the best resistor for the job was the 10,000 ohm resistor. This allowed for the circuit to detect the light about 3 metres away from the robot and allowed for ample sensitivity closer to the light source, allowing the robot to accurately place itself 20 centimeters away from the "fire".

2 Software-Hardware Integration

2.1 Higher-Level Operation

To integrate the software and hardware, two finite-state machines (FSM) were used, one nested within the other. The higher-level FSM operated between three states. Initialising, running and stopped. This FSM was part of the design to initialise the sensors and actuators so that they could be used in the running state. Another feature of this FSM is that it protects the battery of the robot. More details about this FSM can be seen in Figure 2.1.

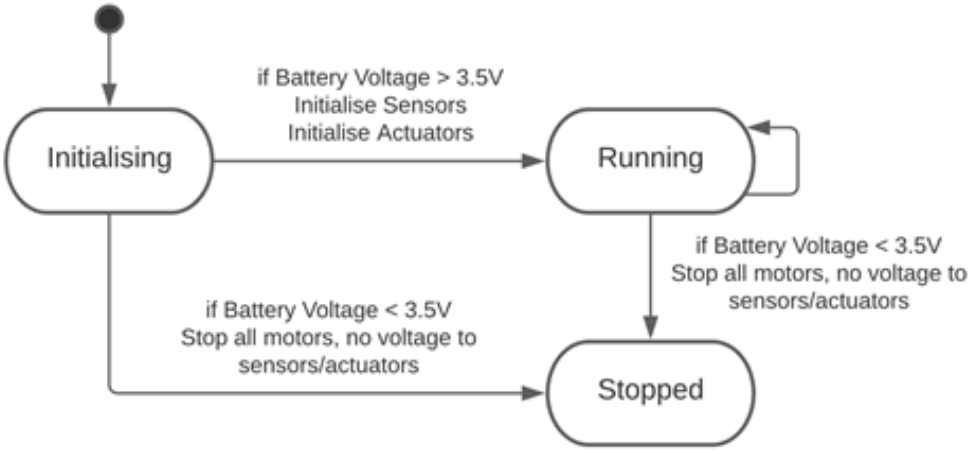

Figure 2.1: Higher level finite-state machine diagram

2.2 Running State Operation

Nested within the running state of the previous FSM, is a lower-level machine that carries out the completion of the project task. This machine has five states. A start and stop state, which initialise and complete the run, as well as three main states called find, blow, and avoid (as seen in the finite-state diagram, Figure 2.2).

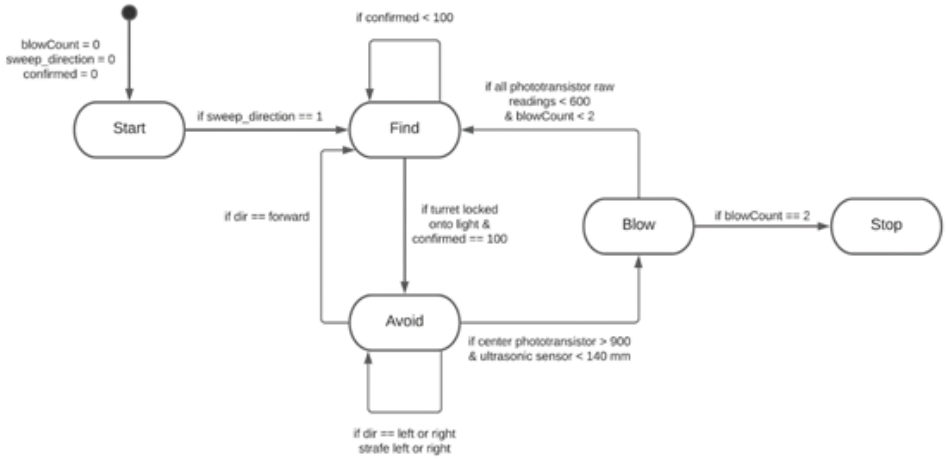

Figure 2.2: Finite-state machine nested within running state

The start state rotates the turret motor along its full range, looking for the highest raw phototransistor value and remembering the position. The state then moves the turret to that

position, and then transitions to the find state. The idea behind this initial state is that the robot will travel to the light that is the closest, rather than the first light it sees in the find state. This should decrease the time of the entire run as in some cases, it will reduce the distance travelled to the first fire being extinguished.

The next state, find, uses the turret motor, three phototransistors, and the four VEX Motors to align the turret with the first light it sees, and then rotate the chassis to be facing the same direction as the turret motor. There is then a 100ms validation period (controlled by the integer variable, confirmed) to stop the chassis from overshooting the alignment. More details on how the motion of the turret and chassis is controlled is discussed later in section 4. Once the turret and chassis are aligned, the state transitions into the avoid state.

The avoid state makes use of fuzzy logic control, the ultrasonic sensor, and the IR sensors on the robot to avoid obstacles on the table. The logic behind how these pieces of hardware and software work together is discussed in detail in section 3. To make use of the fuzzy controller, the avoid state uses the direction determined from the controller to determine the required reaction from the robot. These reactions are travelling forward, strafing left or strafing right. If the robot strafes, it will then travel forward until it sees another obstacle. If that new obstacle is seen as a light from the phototransistor, the robot will enter the blow state. Otherwise, the find state is re-entered.

The blow state first aligns the turret motor and chassis with the light, before moving from 140mm away to 100mm away from the light with assistance from the find function that is used in the find state. The fan is then turned on until all the phototransistors have a raw ADC reading of less than 600. Once this condition is satisfied, the robot reverses for a 300ms period and the integer value blowCount is incremented. The next state change is dependent on the value of blowCount. If blowCount is equal to 2, the next state entered will be the stop state. Otherwise, the robot re-enters the find state. One problem incurred with the blow state was that the blowCount incremented when not necessary. To fix this, a 250ms period was added after checking if all phototransistor ADC values were below 600. After this 250ms period, the same condition is checked again before incrementing blowCount, to ensure that the incrementation is valid. The final state, the stop state, simply stops all of the actuators on the robot once the state is entered from the blow state.

2.3 Additional Design Features

The functionality of the robot in this project, unlike the previous project, is almost purely reactionary based control. In saying this, the accuracy of the robots movement did not need to be as high as it was implemented in project 1. With this in mind, more timer based open-loop control was used in this project when controlling small actions such as when reversing after extinguishing "fires" and strafing away from obstacles, where more advanced control was not necessary.

This also meant that the robot did not have to show any accuracy in speed when travelling around the table. Therefore, for general forward, backward and strafing movements, a constant motor speed was used. The size of the PWM pulse controlling the motor speed was set so that the robot could take a cautious approach around the board, leaving enough time for the robot to react to obstacles and lights in its way, but also complete the course within the 60 seconds for the maximum timing grade.

3 Obstacle Avoidance - Fuzzy Logic

3.1 Problem Description

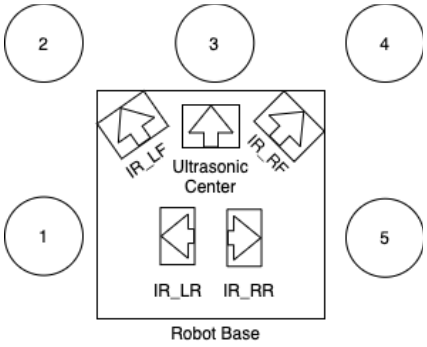

Figure 3.1: Obstacle Case

To get to the fires that must be blown out, the fire-fighting robot must navigate a bounded table containing 4 cylindrical obstacles. These obstacles are of the same dimensions as the two structures containing the fires to be extinguished (105 mm in diameter, 100 mm high). Every collision with either the table wall or an obstacle results in a deduction of one mark, thus collisions are to be avoided.

The brief restricts the placement of the obstacles such that the distance between any two obstacles (including the wall) must be large enough for the robot to pass through in any orientation. From this, it is assumed that the robot will be able to move through any gap simply by strafing (i.e. the robot does not have to turn when avoiding an obstacle).

Figure 3.1 shows the possible positions of obstacles (could also be the wall) around the robot. Although it is possible for obstacles to exist in the blind spots of the robot (e.g. between obstacles 1 and 2), it is assumed that the sensor configuration discussed in Section 1 provides enough coverage to see all nearby obstacles that might need to be avoided.

3.2 Obstacle Avoidance Design

To reduce the complexity of movement control, the functions for discrete movement directions (forward and strafe right/left) developed in the previous project are re-used. These functions were designed by considering the mecanum wheel kinematics and deriving the voltages required to generate a specific motion (see Project 1 report, Section 1 for a detailed explanation). Using these functions, only 1 of 3 possible actions can be taken at any point. For this reason, it is very easy to use a switch case to select which movement should be selected. However, although more difficult, a modified fuzzy logic approach was taken for obstacle avoidance to create a challenge and stimulate learning.

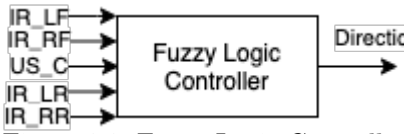

Figure 3.2: Fuzzy Logic Controller

Figure 3.2 shows the inputs and outputs to the fuzzy logic controller (FLC). The inputs prefaced with "IR" are distance readings from the corresponding infra-red sensors seen in Figure 3.1, while US˙C is a distance reading of the ultrasonic sensor placed at the centre front of the robot as seen in the same Figure.

To get the distance readings in mm, the Sensor.cpp class developed in the previous project (see Project 1 report, Section 3) is used. This class filters the input signals from the IR and US sensors and converts those readings into mm distance readings using calibrated functions (see Project 1 report, Section 2). The direction passed out at the end of the controller is a discreet "0" (forward), "1" (right) or "-1" (left) which is used in a switch case to determine which actuation function to send to the robot.

3.3 Fuzzification

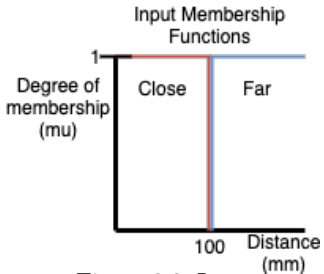

Figure 3.3: Input Membership Functions

To convert the distance inputs into fuzzy variables, two input fuzzy sets are defined: "close" and "far". As seen in Figure 3.3, there is a simple threshold of 100 mm (selected to place the robot's center exactly 200 mm away from a fire if this obstacle is actually a fire) that divides 0% and 100% membership to the close and far sets. It was initially intended that these membership functions be altered to include a region of linear overlap of about 25-50 mm. However, through testing it was found that the best performance was achieved with a simple threshold switch. This means that the membership functions have precise values (not really fuzzy) and are essentially just flags, hence why this

approach could be implemented easily with only if-else statements. The approach taken has the advantage of being more easily extensible however, should it be determined that overlap in the fuzzy membership functions is beneficial.

3.4 Inference and Aggregation

Once the inputs are converted into fuzzy values, it is easy to use fuzzy rules. Table II shows the three rules used to determine the value of the output membership sets (see Section 3.2.3). Inference is performed by taking the minimum value of the antecedents for each rule to set as the corresponding consequent. As there is only one rule for each output set, aggregation is simply performed by taking the maximum of each consequent and zero for each rule. Although this step doesn't achieve anything with the current ruleset, it is still performed to make it easier to add other rules later if required.

Table II: Fuzzy Rules (note d.c. = don't care)

	Antecedent					Consequent
Rule	*IR_LF*	*US_C*	*IR_RF*	*IR_LR*	*IR_RR*	Direction
A	d.c.	Far	d.c.	d.c.	d.c.	Forward
B	Close	d.c.	Far	d.c.	Far	Right
C	Far	d.c.	Close	Far	d.c.	Left

3.5 Defuzzification

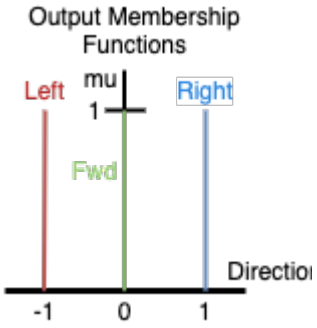

Figure 3.4: Output Membership Funct

Three output sets are defined which are assigned to in inference and aggregation (see Table 3.1). These sets are "left", "forward" (fwd) and "right" and define the level of support for each possible direction of travel for the robot.

Figure 3.4 shows how the output membership functions map to the direction output. As the direction output is a discrete value, the defuzzification process is very simple: select the output membership function with the greatest value from aggregation, this defines the direction output.

3.6 Results

Section 2 describes how the robot changes into and out of the obstacle avoidance state in a higher-level finite state machine. When in the obstacle avoidance state, it is found that the robot is able to successfully avoid collision with the wall and the obstacles simply by strafing. Occasionally an obstacle will fall into one of the robot's blind spots (between obstacles 1 and 2 or 4 and 5 in Figure 3.1) thus leading to a slight skimming of the obstacle with a wheel, or strafing into an unseen obstacle. Due to the limited number of sensors provided, and their limited field of view, this is unavoidable. With the obstacle placement restrictions defined in the brief, cases such as this are very rare.

As the robot's rotation is governed by the "find light" state (see Section 4), when the robot rotates, it does not use the above obstacle avoidance logic. This can lead to collisions with the wall while turning if the robot is forced into a tight gap near the wall. It can also lead to moving an obstacle into a blindspot thus also resulting in collision. These cases are very rare however, and again almost never occur with the restrictions in the brief. Overall, the obstacle avoidance algorithm developed works almost flawlessly over a variety of situations and thus allows the robot to successfully avoid obstacles while attempting to reach the target fires.

4 Fire Sensing and Extinguishing

4.1 Fire Sensing

The approach taken to sense the fire was to track and lock onto the fire with the turret, while attempting to minimise the angle difference between the turret and the base of the robot.

To sense the fire, three photo transistors were positioned at the height of fires. This allowed the ability to compare and hone in on the position of the fire by comparing the left and right photo transistor values with the center reading. The remainder of the turret consisted of the fan and blowing shroud to better direct airflow. This whole assembly was mounted on a servo to achieve a 150 degree field of vision for the robot.

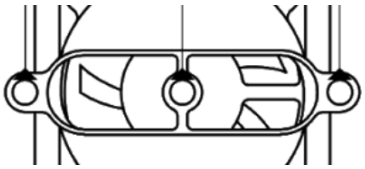

Figure 4.1: Photo Transistor Placement on Turret

To minimise the noise of the circuit, a running average filter was applied to all the phototransistors and a threshold was defined to eliminate any false positives. The logic worked on the following boolean logic:

Table 4.2: Boolean Operator Logic

Boolean Name	**Logic**
Left	$Left > Center\ or\ Only\ Left$
Right	$Right > Center\ or\ Only\ Right$
Center	$Right < Center\ and\ Left < Center$ Or $Right > Center\ and\ Left > Center$
Sweep	No Lights Detected

These Boolean operations then determined how to position the turret on the robot.

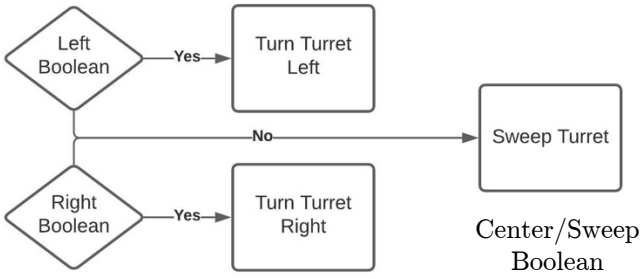

Table 4.2: Turret/Robot Position Flowchart

While the turret is locking on to the fire, a control system is running simultaneously to minimise the angle difference between the angle of the turret and the robot as illustrated below in figure.

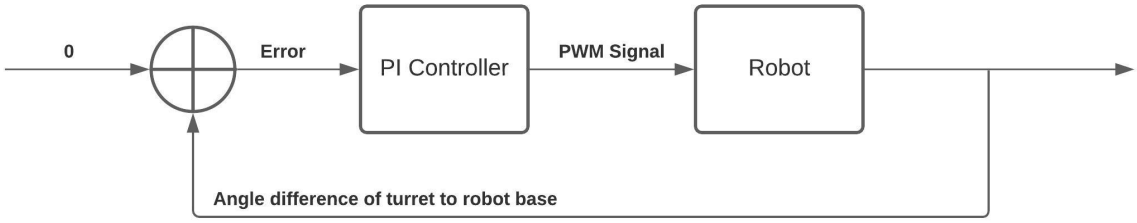

Figure 4.3: Robot Base Position Fire Tracking Controller

Once the robot becomes lined up within 14 degrees of the nearest fire, a confirmation period of 100 iterations begins which allows the robot to transition into the avoid obstacle state to approach the fire.

4.1.1 Initial Sweep

When the robot is initially started, an initial sweep first occurs while the robot is stationary to find the nearest fire to the robot. It achieves this by first sweeping the full range of the turret while recording the highest encountered raw photo transistor value along with the angle of the turret when it saw that value.

Upon a full sweep, the turret can then return to this position where the motors can be engaged and begin aligning the base with the direction of the turret.

Subsequent sweeps will always start with the motors engaged as with a known constraint of only being 2 fires to extinguish, and having already found the nearest one, the second fire will be the only destination the robot should aim towards going to.

4.1.2 Limitations of the Turret

To achieve a high field of view, while keeping the turret safe within the bounds of the robot, minimum and maximum bounds were set which effectively limited the turret's range to 130 degrees. However, as the photo transistors were able to detect fires with a 10 degree tolerance, an effective 150 degree field of view was realised. If no fires are detected within the first sweep, the robot will then start rotating clockwise with the turret fixed at the maximum position until a fire is picked up.

Given the known environment, and through testing, the system implemented on our robot can locate a fire from anywhere within the environment, even if the robot and fire are on diagonally opposite corners from each other.

4.2 Fire Extinguishing

When the avoid state detects that the light is within 20cm of the robot, the FSM transitions to the blow state where a digital high is sent to the MOSFET. This enables the fan to begin extinguishing the fire.

The find state has 2 modes of which it can operate in which dictate whether the robot is moving should move to align with the turret or not. In this case, the find state is called with the robot stationary which allows the turret to quickly and precisely line itself up with the fire.

When all phototransistors no longer see the fire, a successful extinguishing event is recorded and the blow count is incremented. The fan is then shut off.

5 Conclusions

To navigate the table and extinguish the "fires", reaction based control was required for large portions of this project, with a mix of open and closed loop control being used for other aspects of the design. The following conclusions were drawn from the design carried out to complete this project:

- The robot was implemented using a simple aluminium framing chassis, omni-directional mecanum wheels, four modified VEX servo motors, one servo motor to control the fan and an Arduino microcontroller with a pin extension shield.
- Use of the Engineering Departments 3D printing facilities was necessary for extra development of parts such as the fan shroud, in order to place the sensors in the correct position on the robot.
- To control the actions that the robot took, a finite-state machine was necessary in the running state so that the robot could seamlessly and accurately shift between find, avoid and blow states throughout the course.
- A fuzzy logic controller was used in the avoid state, which made use of 4 IR sensors and one ultrasonic sensor. Two fuzzy sets of close and far were used to determine an output direction from the controller, either left, right or forward.
- A turret locking algorithm was written to find and track the fires in relation to the rotational position of the robot. This allowed for the robot to precisely align with the nearest fire at any position within the environment.

The final demonstration went perfectly, with the robot avoiding all obstacles, extinguishing all of the "fires" and completing the course in 46 seconds, well under the 60 second goal for full timing marks. This is an improvement on the previous project, so we are very pleased with this result.