Automated Metal Detection Robotics

By

Jack Li

Sumukh Kenkere Vasudeva Murthy

Final Report for ECE 445, Senior Design, Spring 2022

TA: Akshatkumar Sanatbhai Sanghvi

04 May 2022

Project No. 57

Abstract

Detecting specific metallic objects is a division of automation that has been seldomly explored. In order to create such automation to benefit individual customers for efficiency and industries for safety purposes, we have built a robot using a combination of a metal detector, a microcontroller, ultrasonic sensors, a raspberry pi and a camera which provides a low-cost solution to aid metallic object detection and identification. Such robot could ideally navigate a given room in a short span of time with the help of the pre-programmed microcontroller while simultaneously detecting metallic objects with the sound sensor and identifying them using the camera. Our results show a promising application that be exploited in the future to further automate robotics for metal detection purposes and potentially collaborate with other robotics design for developing multi-functional robotics.

Contents

1. Introduction

Metal detection has been a prevalent technique that is useful in various areas of application: individual personnel use this technique to search for lost items like keys; industries can thus probe important engineering components; there is also on-going attempt to develop metal detection's application for spacecrafts and sample cultivation[1]. But various issues, which may be small or significant, are associated with each of these applications. For individual use, time consumption is worried by many potential customers and a more convenient way of searching for lost items may be inquired; for industrial purposes, safety is a significant concern, and people cannot always enter construction sites themselves to search for those components as accident happens more frequently on the site; for other space exploration purposes, how precise the signals can be sent back to Earth, and how would astronauts manipulate these devices safely should also be considered as the space environment is entirely different.

Therefore, to develop a general-purpose solution to these problems, an automated metal detection robot is needed for several reasons: firstly, it can search for targets and avoid obstacles automatically, which minimizes human forces involved, decreasing the amount of time spent on controlling the device; secondly, as human forces now remain out of sites, safety is enhanced, and people now only need to focus on analyzing the data once the robot has finished probing procedures; thirdly, it has the potential to be extended to other robotics, for example, combing several functions of metal detection, radioactive rays receival and transmission together, which could be very helpful in various research, particularly in space exploration. Current study has focused on expanding metal detector robotics' abilities to send data back to human while controlling the device manually[1,2]; some other developments include precise control of the metal detector coil for better detection flexibility in various environments[3]. However, as automation has not been developed thoroughly in previous research, we, thus in this report, focus on our work in the effort to create an automated metal detection robot prototype, which can detect small metal objects with the ability to avoid obstacles and verify the detected object using pattern recognition. Such project, thus, needs to have several high-level functionalities to get the above job done.

1. High efficiency. This is a functionality trait very important to customers individually. A higher efficiency device can reduce the time spent on a particular scheduled task.

2. Safety insurance. This is another high-level functionality that is both crucial to individuals and industries purposes. We want to make sure that the device does not increase more unnecessary dangers to users and their surroundings. Therefore, the device needs to avoid obstacles and stop if any measure taken afterwards would crash itself automatically. Electricity, mechanical parts should be within safety minimum safety limits (for example, the highest operation voltage should be below at most 30 volts).

3. High precision/accuracy. This functionality features another aspect of automation. Previous research focuses on data transmission back to human receivers for data verification. To greatly utilize the full functionality of the device, we expect a model which can do the initial verification step for humans before they need to engage in again (for our particular metal detection purpose, this would mean that it can identify the detected object to either be our target object or not with enough accuracy).

Thus, given these 3 high level functionalities, we have the following general structured design (figure 1), or top-level diagram for the robotics system as our prototype:

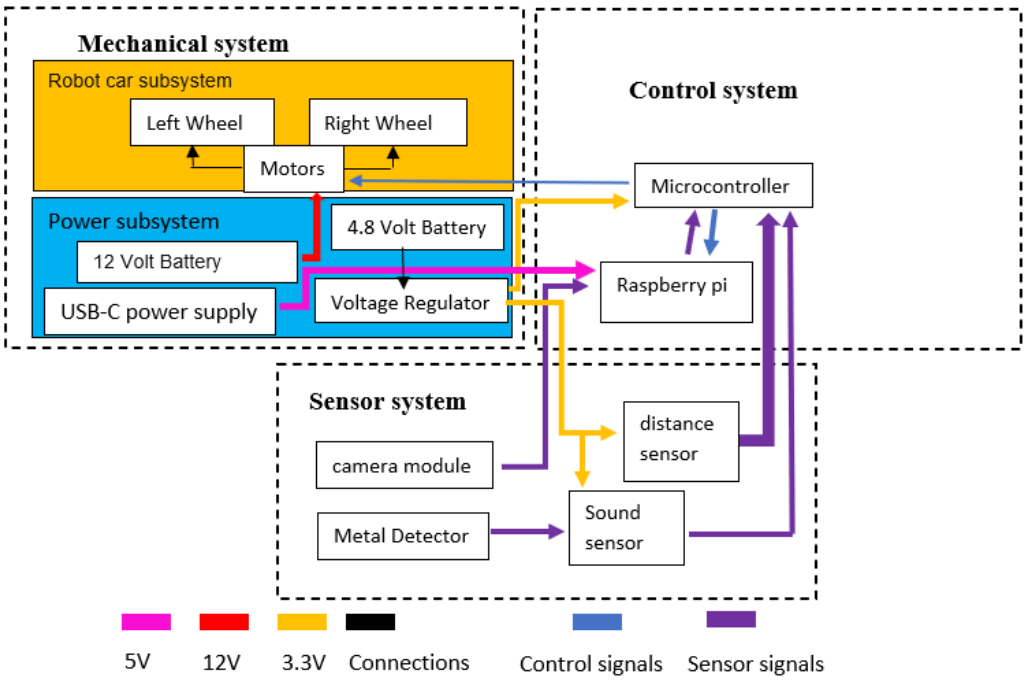


Figure 1: Block diagram of the overall design. The mechanical system consists of power subsystem and robot car subsystem and is responsible for the physical movement of the robot. Control system is centered around Microcontroller where it processes all the sensor signals and generate control signal. Finally, the sensor system consists of distance sensors, sound sensor coupled with metal detector, and camera, which generate input signals to microcontroller.

This prototype can be summarized into 3 systems, each individually is responsible for concentrated tasks: mechanical system, control system, and sensor system. The mechanical system delivers power to the whole robot, including Printed Circuit Board (PCB), Raspberry pi (RP), sensors, and the motors. It also includes a robot car subsystem, which is the entity (or "body") of the robot. We use a simple design for the prototype such that two motors with respectively connected wheels can perform all the motions necessary for the robot to move, turn around, and avoid the obstacles. The control system is the "brain" of the robot and is also the interconnection across the whole robot system web. It receives signals from sensor system, generate control signals, and deliver them to the mechanical system to drive the car in specific operations. Lastly, we have the sensor system—the "eye" of the robot. We use distance sensors to detect any potential obstacles, sound sensor coupled with metal detector to detect presence of metals, and camera for general purpose detection. Initially, we would like to use camera as an extra "vision" subsystem to tell the robot some specific operations, for example, whether the obstacle ahead should be avoided or searched (as some obstacles are soft and are actually where most hidden metal items may reside, we need to go "into" them to do a thorough search instead of avoiding it). But as our project progresses with unexpected events happened, we have to simplify the function camera can perform to finish prototyping the project. It ends up functioning as the eye to pattern recognizer, which tells the system whether the objects detected as metals, are the targets. Therefore, we can see how these systems are connected: "eye" tells the "brain" the information around, "brain" synthesizes the information and control the "body" to move.

This is a straightforward and simple description of the whole robot's top-level design and its systems. In the following report, we will go into depth about our design in chapter 2 and verify them in chapter 3. Costs are described in chapter 4, with chapter 5 summarizing our whole project. Overall, we reached a state where nearly all the systems designed work well individually. However, due to time constraints, we have shortage of parts and lack of parameters modifications to optimize the performance of this prototype metal detection robot. We will summarize them and list potential improvements to conclude this project in the end.

2 Design

2.1 Design Procedure

The design procedure can be divided into 2 big components:

1. Subsystem technical design and 2. Selecting parts.

Procedure 1 offers the minimum requirements and variables that can be changed for each subsystem, while procedure 2 provides options and alternatives that satisfy the same requirements obtained from procedure 1. In this section (2.1), we will introduce the most general and basic design choices made during the whole design process. For more detailed descriptions, we have broken up all the subsystems into individual modules and will depict them along with "specific" design alternatives in section 2.2.

The three systems in our design each has minimum requirements. For mechanical system, with our two-driving-wheel design, the motors we choose will need to overcome a torque that is set by the total weight of the system and frictional forces between motors, which is given by:

$$\tau > k_s mgr$$

Where τ is the maximum torque that can be provided from the motor, m is the total mass of the robot car, g is Earth's gravitational acceleration constant, r is the radius of the driving wheel, and k_s is some unknown coefficient of friction existed within the motor-wheel system, for example, non-ideal frontal wheel inner contact that stops every car's motion when driving wheel no longer provides power. This model provides an estimation for the motor torque we need to choose. For mass 5 kg, driving wheel radius 7 cm, and coefficient of friction between wheel's contact with the central axis 0.1 (common for steel-to-steel contact[4] with oils, static coefficient of friction), we have the torque minimum value to be about 3.5 kg·cm. Another factor which is also essential to mechanical system design is the motion speed, which is provided by the motor's Rotations Per Minute (RPM). To achieve a reasonable motion speed, we expect about 0.5 m/s, which corresponds to

$$RPM = \frac{v}{2\pi r} \cong 70$$

To summarize these design considerations for procedure 1 for mechanical system, we need motor torque at least 3.5 kg·cm and RPM about 70. Then upon entering procedure 2, we quickly find a motor with rated torque 4.5 kg·cm and RPM 100 which satisfies this minimum requirement well. There is actually an alternative design for the mechanical system, which is using 2-connected wheels system to drive the robot (like a tank). But this design needs a stronger motor, which in the end will affect RPM (with the same power, a higher torque requirement results in a lower RPM), therefore, we choose this design in the end.

Now for the control system, the baseline design is that it needs to have enough input/output ports (GPIO) to receive all the signals from the sensor system and deliver control signals. To figure out this baseline, as control system and sensor system are connected closely together, we will describe the minimum requirements here collectively. For sensor system, we need at least 3 distance sensors (with each one covering front, left, and right side respectively), and one sound sensor to be bonded to metal detector for synchronous operation. As most sensors have 4-pin connections with 2 pins delivering

interactive signals, plus at least 2 control signals required for the two motors, we need a minimum of 10 GPIO ports for the microcontroller. Considering this minimum requirement with procedure 2, several options become available immediately: ATmega328 has 23 GPIO ports and is compatible with Arduino programming interface; STM32F103C8T6 microcontroller has flexible GPIO ports up to 80, and it has various data input/output types supported. But in the end, we chose RP2040 which is a microcontroller newly manufactured from Raspberry Pi. The 25 GPIO pins satisfies our minimum requirements, and it has python compatible Interactive Development Environment (IDE) which is slightly easier than the two alternatives to program.

Consequently, with RP2040 GPIO voltage range limited to 0-3.3 volts, turning back to sensor system, we have 2 options for the overall design. Firstly, we can use 5-volt sensors and use multiple voltage regulators to convert to 3.3 volt; alternatively, we can use 3.3-volt sensors throughout the system but use a single voltage regulator in the power subsystem. The advantage for the second option is obvious: it has less components required and is cleaner to be implemented. Although the microcontroller will have lower efficiency considering that it has maximum performance at 5 volts, we can always separate the power lines and therefore maintain a good performance.

We, thus, have summarized the design procedures and our most general considerations when designing the subsystems. In the following section, the subsystems are compacted into modules, and we will focus on the design details for each module.

2.2 Modular Design Details

Considering the 3 systems we have and their interconnections, we can break the whole project into 5 modules with 3 hardware modules and 2 software modules:

Hardware modules	Software modules
Peripheral module: includes all the components attached to PCB module (power sources, sensors, Raspberry Pi)	**Microcontroller control module**: software program used to control all the signals in the system (manipulate input, generate output)
Robot module: the robot car entity with wheels and motors, which is the end output module	**Raspberry Pi machine vision module**: software program used to pattern recognize detected object
PCB module: provide circuit that connects the other two modules	

Table 1: a summary of all the modules that construct the 3 systems in the project. Hardware modules are the main electronics and mechanics design components while software modules add automation into the electronics.

Doing this can simplify the whole project's design and we can individually implement and test each module (although some modules require the cooperation of other modules to test, including microcontroller control module and Robot module). In the following, we will give detailed descriptions for each of these 5 modules, with an emphasis on PCB, microcontroller control, and machine vision module.

2.2.1 Peripheral module and Robot module

The peripheral module and robot module are all external modules which attach to PCB module. Robot module overall is designed by us with the help of the machine shop and can be illustrated in figure 2.

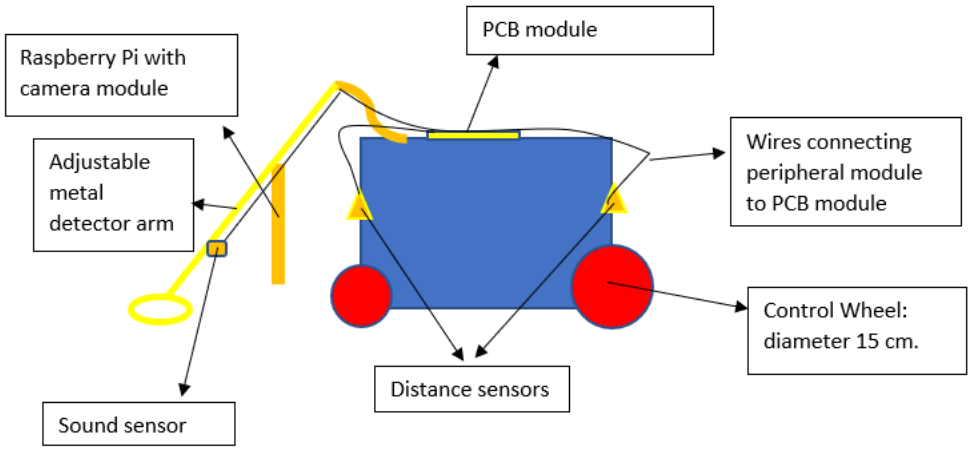


Figure 2: Physical diagram illustrating the robot module. In this figure, PCB module and part of Peripheral module (sound sensor, distance sensor, and Raspberry Pi) are also shown to be interconnected with robot module, thus, demonstrating how this project's final placements for all the components are.

Dimensions are not drawn to scale. The design features a fixed metal detector stretching out from the robot car's main body forward close to ground for detection purposes. Sound sensor is placed near the metal detector such that it will not trigger metal detector's detection (as sound sensor is also metal) while keeping a reasonable distance to be sensitive enough to collect sound from metal detector. Distance sensors, depending on the design purposes, are mostly placed in the front, left, and right side of the robot to gain a thorough vision of its surroundings. If the robot can move backwards, then an extra one can be placed at the back side, but in our design, we recoil from such placements which will be discussed in more detail for the reason in section 2.2.2. PCB module is placed around the center for easy connections through wires to the other modules. Lastly, although not shown directly, motors are fixed with the wheels at the backside of the car to drive the car's motion. As discussed in section 2.1, we chose 2-wheel driving design because it can accomplish the desired motion operations while reducing power required. More details about the car's operations will be discussed in section 2.2.2 and 2.2.3.

Peripheral module is relatively more straightforward. These are the components directly associated with PCB and can be summarized in the following table.

Components	function and parameters
12-volt battery	provide power to PCB to drive motors
4.8-volt battery	provide power to PCB to drive microcontroller and sensors. In the PCB module, we will see an extra voltage regulator used to convert it to 3.3-volt source.
ultrasonic sensor	Part number US-100, working voltage range from 2.4 volt to 5.5 volt
sound sensor	KY-037 sound sensor, working voltage range 3.3 to 5 volt

Table 2: a summary of the most essential components with their values and functions for Peripheral module.

As we in the end adapt to 3.3-volt power system for all the electronics (section 2.1), we choose sensors such that they can comfortably work at 3.3 volt without affecting the performance, and thus, US-100 and KY-037 sensors are perfect to be implemented. Working distance range is also an important parameter when designing this module. We need to ensure that distance sensor can have sensitivity at least around 10-20 cm where robot car needs to respond and stop. But this design decision is actually a parameter that hasn't been optimized throughout the project, and therefore is pending to change. With a fully working model, combined with distance sensors, several trials of testing need to be done to ensure that the braking distance is long enough, and unfortunately, we in the end of the project does not have enough time to optimize this parameter. We will discuss more in Chapter 5 conclusions.

2.2.2 Circuit Design—PCB Module

PCB module consists of the PCB only, which is the central circuit we have to connect everything. It essentially has functions related to all 3 systems, including providing power (mechanical system), receiving and interpreting signals (sensor system), and generating control signals for various operations (control system). Figure 3 shows the comprehensive design details of the circuit schematic.

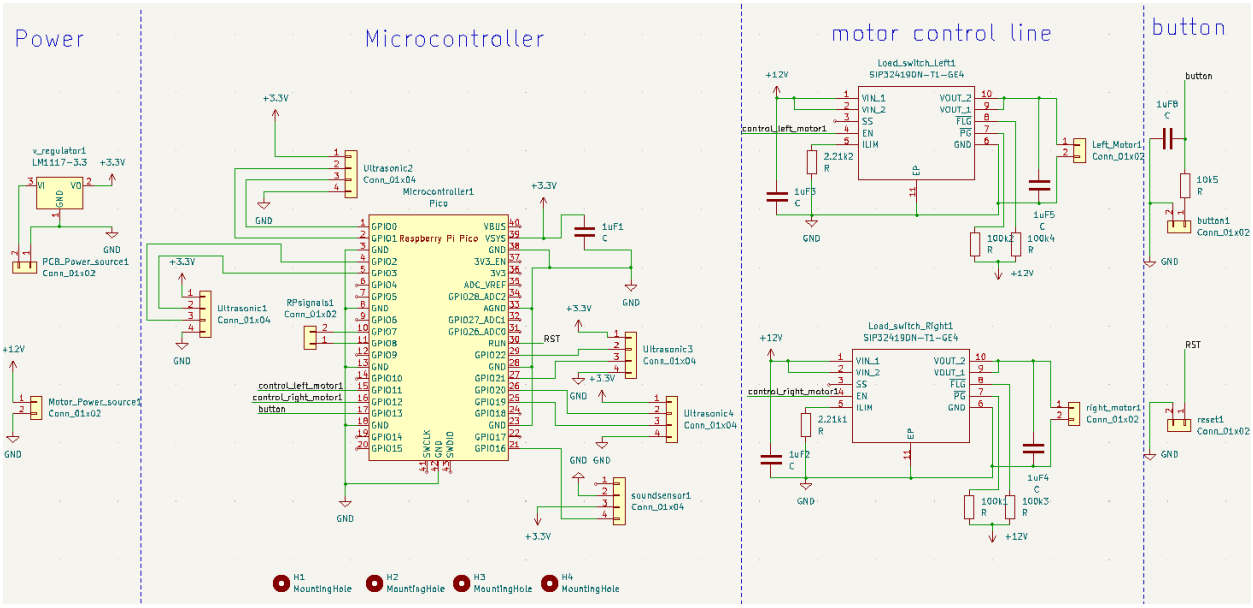


Figure 3: Circuit schematic of the whole PCB module. It has been divided into 4 submodules: power, which connects to the power supplies in Peripheral module; microcontroller, which connects to sensors in Peripheral module, programed by microcontroller control module, and deliver 2 control signals to the next submodule; motor control line, which is a submodule centralized around 2 power switches to convert control signals to real, physical power provided to motors; button, which consists of control signals used to interrupt/initiate/start the whole PCB module.

Power submodule typically uses a simple 2-pin connector to connect external 12-volt power source to the system which is used to drive motors. Another 2-pin connector is coupled with 3.3-volt voltage regulator to provide power to sensors and microcontroller. There are not many design alternatives for this submodule, as in the end, only AMS1117-3.3 voltage regulator is available online and can be ordered. But since it delivers current between 0 and 1 ampere[5] and maximum current capacity (1A) is

much larger than the working current of the microcontroller (about 90 mA[6] in working mode), the regulator would work well with our system theoretically.

Button submodule is the only human force needed for this robot to move. As we have mentioned in the introduction chapter, our goal is to "automate" the metal detection robot. The only button is used to start/stop the car along with an optional reset button to re-initiate the system in case the microcontroller is stuck in an infinite loop. All other operations are performed automatically with the use of the following submodules.

Microcontroller submodule is built to connect all components in Sensor system and Mechanical system (or in terms of modules, the Peripheral module and Robot module). To summarize this submodule, it is easier to show all its pin connections in the following table:

	Pin	function and parameters
Input	GPIO 0-1	Send trigger and receive echo signals from ultrasonic sensor 2
	GPIO 2-3	Send trigger and receive echo signals from ultrasonic sensor 1
	GPIO 7-8	Send trigger signal to Raspberry Pi by setting it high and receive Raspberry Pi return signal (high/low)
	GPIO 16	Receive digital output signal from sound sensor
	GPIO 19-20	Send trigger and receive echo signals from ultrasonic sensor 3
	GPIO 21-22	Back up port for the fourth ultrasonic sensor
	GPIO 13	Receive signal from button. When low, initiating the system
output	GPIO 11-12	Send two control signals to motor control line submodule to control the motion of the robot car
	Vsys	1.8~5.5 volt. In our setup, we use 3.3 volt generally for microcontroller power.
	GND	All ground pins should be grounded for proper operation.

Table 3: a summary of the microcontroller submodule with pin assignment.

For this hardware configuration, we only list the main connections, parameters, and basic operations. Detailed operations the microcontroller can perform is introduced in section 2.2.3 for software side's descriptions. Typically, this is the minimum design choices we made: monitor at least 3 distance sensors for forward, lefthand, righthand direction information; monitor one sound sensor's digital output value for metal detection purposes; send 2 control signals with each responsible for one side of the motors. Overall, the heart of the design is to figure out how those 2 control signals can be interpreted by the PCB and generate physical power for the motors, and this is how our next submodule—motor control line—works.

Motor control line submodule relies on the design of power switch and is the heart of our PCB module design. In the simplest description, it contains a power input pin, an output pin, and an enable pin to control the power flow from input to output. Minimum requirements we need to consider is working voltage (with our 12-volt motor, at least 12 volts should be allowed) and maximum allowed current (our motor works at 1 ampere rated current, therefore at least 1 ampere for I_{max}). Under this condition, a lot of power switches are satisfactory: TPS22810 manufactured by Texas Instrument, STELPD01 manufactured by STMicroelectronics, and the one we use in our final setup SiP32419 which has wide range of working voltage from 0 to 28 volt and controlled maximum current by using a resistor. Although in our final setup, two power switches are used individually to control each motor, there are actually several design

alternatives we had for this submodule. Initially, we wanted to achieve a maximum number of operations the robot car can perform, and we had 4 control signals (2 for each motor), as can be shown in figure 4 along with the table for operation summary.

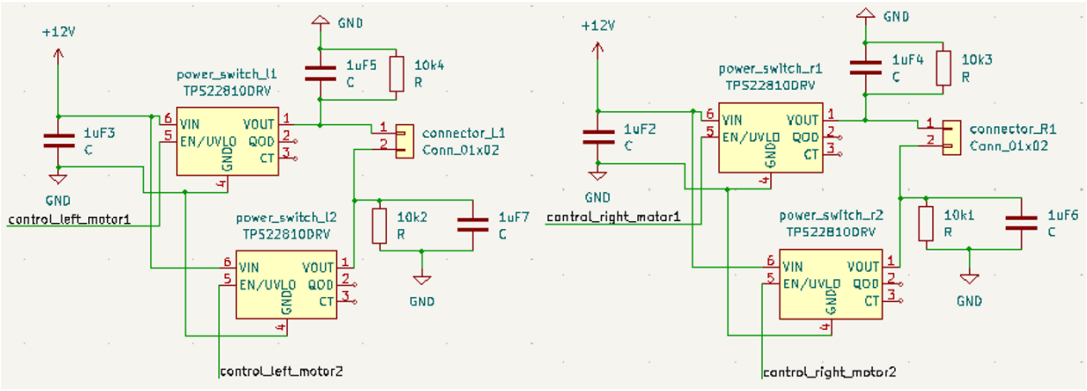


Figure 4: Original design of Power control line submodule. Each motor is controlled by 2 signals instead of 1 which we used later.

operation	motor control left 1	motor control left 2	motor control right 1	motor control right 2
Forward	1	0	1	0
Backward	0	1	0	1
Turn left	0	0	1	0
Turn right	1	0	0	0
rotate left	0	1	1	0
rotate right	1	0	0	1
stop	0	0	0	0

Table 4: Operations the robot car can perform ideally with the original design. By changing the various control signals to be high or low, we can perform a total of 7 operations which are ideal to control our robot to move in various ways under different environmental conditions.

But there is a severe issue with this design which we figured out later. Although initially we wanted to control the polarity of the motor (run in forward/backward direction) by using control signals such that when one terminal of the connector is connected to power (power switch on) while the other terminal is connected to ground via “a resistor” (power switch close), it turns out that this is equivalent to a voltage divider circuit with the “motor”. Doing this will result in a large amount of power loss if using a small resistor (typically should only be about a few ohms), or it will stop the motion at all because the voltage is divided to be only a few decimals of volts to motor.

Upon figuring this out, there is still more design alternative, which may ideally fix this issue once and for all while keeping the same number of operations to perform. The problem in essence is that we need to get the connectors freely connected to either ground or 12 V power supply. But to do this, this would mean both 12-volt source and Ground will share the same power line, how can we possibly do that? The answer is hidden within the number of power switches used. If we can connect the two lines (power and ground) in parallel, with one line connecting to the output of the first power switch with 12 volts at the input, while the other line connecting to the “input” of another power switch with the output

connecting to ground, then we can theoretically create a bi-directional power line for the connector, and thus, can control the motor to rotate in both forward and backward directions, which can be illustrated in figure 5 using 4 power switches for “each motor”.

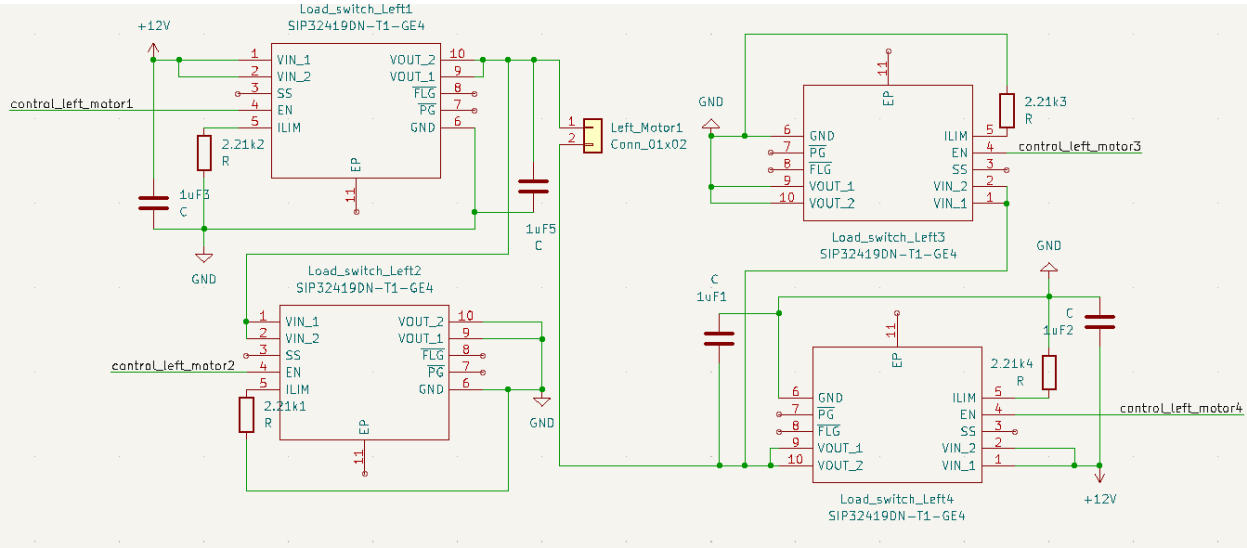


Figure 5: Design alternative of Power control line submodule. Each motor is controlled by 4 signals, and we need a total of 8 control signals (power switches) to accomplish all the operation desired for the robot car.

operation	motor control left 1	motor control left 2	motor control left 3	motor control left 4
Forward	1	0	1	0
Backward	0	1	0	1
stop	0	1	1	0

Table 5: Operations a “single motor” can perform with the four control signals under this design alternative. When signal 1 and 3 are high and the rest low, connector port 1 is connected to 12-volt power supply and port 2 is connected to ground with no bias resistor. To run the motor in reverse direction, we just reverse the role in each terminal by setting signal 2 and 4 high instead. Lastly, to stop the motor, both terminals are connected to ground via control signals 2 and 3 setting to high.

This is the ideal design that we can potentially have and would maximize the performance of our robot car to accomplish all the operations as shown in table 4. But unfortunately, due to shortage of time and components, we were not able to implement this design. Considering the complexity associated, and that new PCB order would not arrive in time in case minor error happens, we have to abandon this design alternative for logistical reasons. It would be an interesting next step to do to advance our project. Therefore, in the end, we used the design as shown in figure 3 for initial testification. But overall, such design is able to demonstrate the potential of implementing the more complicated design alternative, as will be described in chapter 3, these power switches can pass the verification and successfully drive the motors. The simplified design also has enough operations for the robot car to complete the tasks we expect, although it now relies more on the control program to avoid obstacles, which we will discuss in the next section.

2.2.3 Microcontroller Control Module

This is the central software module used to run the whole robot using the microcontroller. Design has several considerations: (a) How to move the robot to gain maximum detection range? (b) How to ensure safety in case the robot is close to obstacles? (c) How to increase efficiency and prevent robot from repeating things constantly?

To begin with, we use the firmware called MicroPython to program the device, which is a Python IDE specifically designed for microcontroller. Now generally, in order to answer the design questions above, a state machine is designed to evaluate the robot's operations.

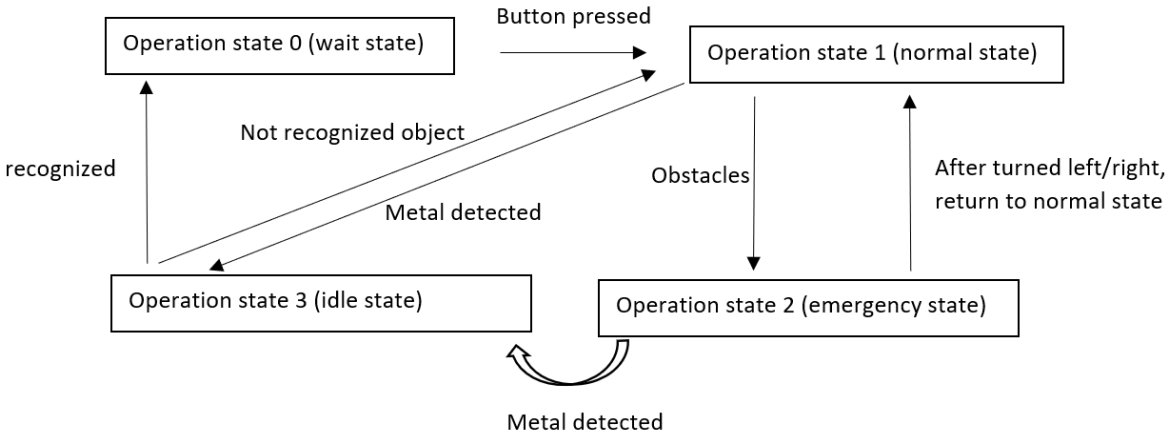


Figure 6: State machine illustrating the central design of microcontroller control module. Operation state 0 and 3 are considered "stop state", as during these two states, the robot itself will not move. Operation state 1 is considered "safe state", during which, the robot will periodically perform tasks associated with some "sub-states" (go straight/turn left/turn right, etc.). Lastly, during operation state 2, the robot needs to react to some events (close to obstacles/walls) to ensure safety.

The wait state is an infinite loop which can only be interrupted by signal from button. When trapped within the loop, the robot car will not do anything (motor is stopped) until human user engages to press the button and pull the button pin to active low. Then, the machine enters state 1 (normal state). During this state, robot will perform tasks based on the sub-state machine.

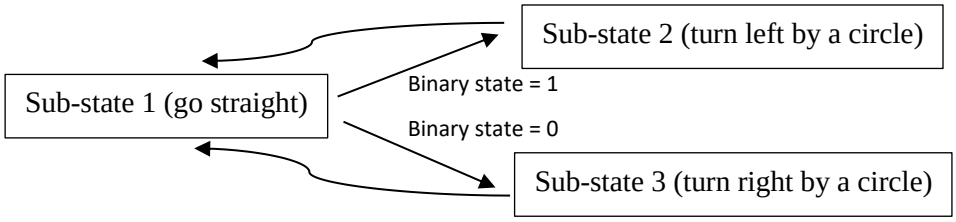
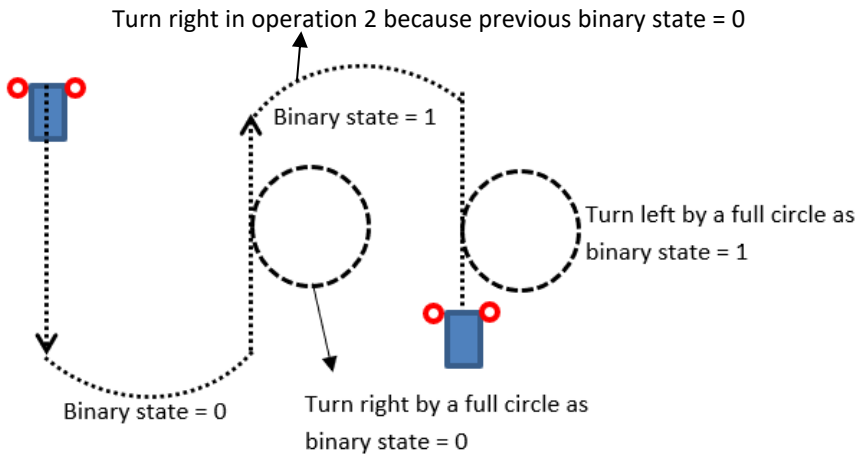


Figure 7: sub-state machine used in operation state 1. Binary state is a special variable which records the "previous" (turn left/turn right) operation performed in operation state 2: if turned left, binary state = 0.

And here we have answered question (a) and (c). The substates 2 and 3 let the robot go around itself by a full circle to thoroughly search its surroundings (go straight, turn around by a circle, and then repeat), which is one way to optimize solution to question (a). The ideal way is actually "rotating by a circle" which requires one motor to be forward while the other to be backward. But as mentioned, in the end, the motor control line submodule is simplified, which now has some slight disadvantages. To answer question (c), the trick is hidden within the variable "binary state". Binary state is high when the previous operation performed in operation state "2" is "turn right" and low the other way around. This essentially



records the information of the robot's route taken and is also used in operation state 2 to decide whether to turn left or right to avoid obstacles ahead.

Figure 8: An illustration of how binary state works in the module. In principle, the car will only turn left when binary state is 1 and turn right the other way around, regardless of if it is in operation state 1 or 2. As shown in the figure, binary state also changes upon "turning left/right" to become "0/1".

Thus, in this way, we can make sure that the robot car will not easily repeat the route it has already taken and increase the efficiency of detection.

To answer question (b), the answers are all contained in operation 2 design. Operation 2 can only be entered when 2 cases happen: 1. There is obstacle ahead and need to avoid. 2. The two sides of the car are too close to walls nearby. For case 1, upon entering operation 2, the car will either turn left or right by half a circle depending on the binary state as shown in figure 8. For case 2, depending on which side has larger free space left, the car will move further outwards towards that direction. Finally, regardless of operation state 1 or 2, if the car gets too close to an object (<20 cm), the system will be back to state 0, and stops the car to ensure safety. Therefore, these design details can actually attempt to optimize the solutions to all the questions we have raised initially during design process.

Lastly, operation state 3 is only used when metal is detected, at which point, the system will begin monitoring signals from the machine vision module. That module will use a camera to see if the detected object is the correct object and send respective high/low signals to microcontroller. If verified, system is back to state 0 waiting to perform the next operation; if not, system is pushed to state 1 and continue to search for another object.

2.2.4 Control System Design–Machine Vision Module

The Machine vision module is built using a raspberry Pi and the Arducam Camera which is plugged into the raspberry pi. Overall, the modules use a combination of image and signal processing.

The Raspberry Pi device and the Arducam are on permanently in an idle state and are activated upon receiving a signal from the raspberry Pi. This is a signal of a logic '1' which is equivalent to a voltage spike of 3.3 volts to the GPIO pin of the raspberry Pi. This signal wakes the Pi from the idle state and initiates the finite state machine (FSM) to process and identify the image. The finite state machine is illustrated in figure 9 which will help simplify how the states run to produce a justifiable output.

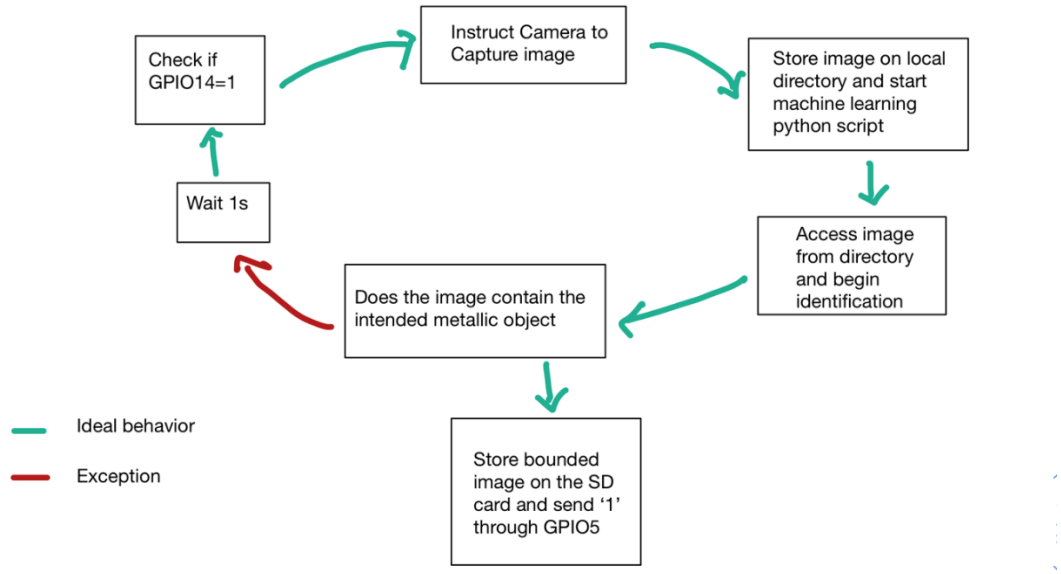


Figure 9: The Finite state machine behind the Functioning of the machine vision module.

Each sequence in the finite state machine is explained in detail as shown below.

1. The Raspberry Pi (Pi) receives a digital signal of '1' on GPIO pin 05 or pin '29' on the Pi and this signal is stored into a local register which the code reads at a frequency of 1s. We have chosen '1s' due to account for the processing delays in the previous modules and the ability of the raspberry pi to conserve power by reducing the clocking speed.
2. After a value of '1' has been confirmed by the python script, the script initiates the section of the code that directs the Arducam to capture an image and store it on the local directory on board the SD card.
3. The stored image is then accessed by the machine learning section of the code which takes the image captured as an input, resizes the image into a convenient size of (608,608,3) which stands for the dimensions of the image which is 608 pixels × 608 pixels × 3 color channels (RGB).
4. The image taken is set up to be run on the pi by invoking a section of code to initiate processing of the image along with identifying the object located in the image. A bounded box containing the recognized image is stored in a separate directory which contains all detected objects. In the case

that the image is the image contains the object which we are looking for, the program continues to step ‘6’.

5. This step pertains to the case that the object detected is not the object we are looking for. In this case, the object’s image is still stored locally on the Pi with the exception that the pi sends a signal back to the Control subsystem with a digital ‘0’ or a 0V output which is the same as no signal being sent. As mentioned before, the Finite state machine of the microcontroller control module waits for a period of 40 s for an input of a digital ‘1’ or 3.3V input from GPIO pin ‘12’ before continuing to look for metals. In this case, we send a logic ‘0’ which means that in the period of 40s that the microcontroller waits for a signal, there is no input and the FSM of the microcontroller registers that the required object has not been confirmed. In this case, the robot continues to look for objects and the pi is suspended into an idle state and waits for the next interrupt from the microcontroller. Upon receiving the input of ‘1’ again, steps 1-4 repeat until the python script confirms the required object has been detected. In this case, the final stages of the Machine vision Module are initiated.
6. When the required object let’s say a fork is detected, the raspberry Pi immediately logs the time and sends a digital ‘1’ or a 3.3 V output to the microcontroller which acts as an interrupt signal confirming the detection of the required object. Then, the raspberry pi goes into a sleep mode for a period that can be customized for example, we intended to use a period of 40 s to ensure that no signals were sent to the microcontroller to prevent random interrupts.

3. Design Verification

In chapter 2, we have presented all the design procedures and details. Note that some design requirements have been explicitly stated already, therefore, in this chapter, some repetitions may be observed. Overall, we would like to divide this chapter into 2 sections: module verification and high-level verification. Module verification provides requirements, verification procedures, and results for each important component in each module. Most of them are summarized directly in **appendix A** and interested readers can directly go to the appendix and see the standard verifications we have performed. High-level verification is, on the other hand, more qualitative, and given our restriction of time, it is not fully verified. But we will still discuss requirements and verification procedures that may be performed.

3.1 Module verification

We have seen in chapter 2 that the whole system can be divided into 5 modules: Peripheral, PCB, Robot, microcontroller control, and machine vision module. Among these 5 modules, only part of Peripheral and PCB modules can be verified independent of the other modules. The other modules, especially Robot and microcontroller control modules, need to be verified together with PCB module at least. In the following, we will describe standard requirements and verifications for the two independent modules primarily.

Let us consider a question firstly: if we want our system to work, what do we need? The answer can be simplified to: (a) electrical parameters are within normal limits (b) system does what we want. The former one is quantitative, while the latter one depends on "what we want". Following this, we now begin with Peripheral module: powers and sensors. From (a), it requires these Peripherals to have "expected" working voltage (and current in some special cases). Depending on the situations, tolerances for the working voltage can differ. Typically, we want the power source to fluctuate within 0.5 to 1 volt for non-voltage-regulated device, and sensor's working voltage can fluctuate as long as their physical output is within the range endurable by the microcontroller, in this case, smaller than 3.3 volt and higher than 1.8 volt (the high voltage level interpretation). To verify these, detailed procedures are provided in the appendix as mentioned, and the results can be neatly summarized in table 6.

Part to verify	**Technical Result (working voltage)**
PCB battery (4.8 v)	4.79 volt to 5.02 volt
motor battery (12 v)	12.9 volt when stabilized
sound sensor	2.7 volt at $V_{digital\ output}$ pin when tested with 3.3-volt power supply
ultrasonic sensor	N/A

Table 6: technical result for Peripheral module

As we can see, all the voltage results are within respective tolerance range. Ultrasonic sensor is hard to measure on the other hand. We are interested in its echo pin voltage, but it is a pulse signal, which cannot be directly measured with multimeter even if we could continuously trigger this pulse signal. An oscillator may be a good option to see its signal but again, since it is not periodic, it is very hard to control the measurement. Alternatively, we use requirement (b) to evaluate this component. The ultrasonic sensor measures distance, and we can directly provide programs to it and see the measured distance in numbers. Or more importantly, we can set important threshold numbers (in our case, 20 cm and 50 cm) in distance, and see if an external signal can change (for example, in section 2.2.3, we should observe the state transit

from 1 to 0 or 2 at the respective distance). Our result in the end shows that the state indeed transits near the boundaries, which is indicated by the braking of the motors when the distance between the sensor and obstacle is less than 20 cm. Such qualitative result is enough to show ultrasonic sensor's performance, as in the end, we only care about these threshold distance values, and how the system responds to values provided from the distance sensor.

Similarly, from requirement (a), we can propose electrical requirements for PCB module, where the most important parts are voltage regulator and power switch. Results are shown in table 7 with details in appendix.

Part to verify	**Technical Result (working voltage)**
Voltage regulator (3.3 v)	3.27 volt when stabilized
Power switch	12.9 volt at output when switch is close; 0.6 volt when switch is open.

Table 7: technical result for PCB module

The other parts of PCB functions are tested with the combination of microcontroller control module (software) and robot module for requirement postulated from (b). These verifications are actually high-level verification and would be better to be discussed in section 3.2 for simplicity and clarity.

3.2 High-level verification

As our project is more practical rather than numerical (data analysis, measurement-based), most of the high-level verifications and requirements would be less quantitative. But indeed, some parameters are crucial to the high-level design performance, and we would like to introduce these requirements and propose how we can verify them. Given requirement (b) to verify the rest of PCB, microcontroller control, and robot modules, we need verifications achieved in several aspects: 1. The modules work together: when pressing button, robot starts to move; when distance sensor detects close obstacles, the robot stops; when metal detector coupled with sensor detects metal, the robot stops and enters state 3 (section 2.2.3). 2. Algorithm in microcontroller control module works in various environments (quantified by the overall time measured for the robot to search for an object). 3. Detection accuracy is good (quantified by the machine vision module, and typically a "good" accuracy is 80%, meaning that 80% of the verified objects are the targets).

For these 3 high-level requirements, we have verified the first one qualitatively which is the basic indicator of our working prototype. Peripheral module sends data (button trigger, distance, sound) to the PCB module controlled by microcontroller control module to generate control signals for motors, which drive the robot module. In the first verification, these 4 modules' cooperation and operations are demonstrated. Unfortunately, due to lack of time, the rest 2 verifications are not done. Verification 2 needs a totally working model and is very essential to optimizing parameters related to safety (when should the robot stop) and efficiency/accuracy (how often should the robot car rotate by a full circle to search the area more thoroughly). Measuring the time taken to search for the target will be a meaningful verification to accomplish. Lastly, verification 3 depends on both the machine vision module itself and the position of the camera, and since this module does not work in the end, further verification is needed to thoroughly test this high-level requirement.

4. Costs

4.1 Parts

This section estimates the total costs we spend on building this project (parts costs), excluding costs related to machine shop.

Part	Manufacturer	Bulk Purchase Cost ($) per piece	Actual Cost ($) (incurred by our group)
Raspberry Pico	Raspberry Pi foundation	4.50	19
Raspberry Pi 4B	Raspberry Pi foundation	120	179
San Disc 64 Gb micro-SD card	San Disc	14	14
Arducam OV5647	Arducam	9.99	9.99
US-100 distance sensor	Adafruit	14.38	43.14
KY-037 sound sensor	Joy-it	5.98	5.98
NiMH Receiver Battery	Limskey	10.89	10.89
Metal detector	Allsun	30.73	30.73
ML5-12 - 12 Volt Battery	Mighty max	15.99	15.99
DC 12 Volt motor	Greartisan DC	16.34	16.34
Total			345.06

Table 8: Parts Costs

4.2 Schedule

This section lists the schedule of this project production in weekly basis starting from February 28th

week	Jack	Sumukh
28-Feb	Finished design documentation with design review. Will start project beginning next week.	Researched ethical considerations and ideas on how to implement product
7-Mar	Finished circuit schematic design.	Researched Possible components that would permit functionality
14-Mar	Finished first round PCB design, ordered parts	Worked with Jack on creating the FSM to run the robot
21-Mar	Machine shop communication	Worked with machine shop on product design and positioning
28-Mar	Microcontroller programming initiated, edited and ordered second round PCB due to issues in the first round	Researched using Eigenfaces algorithm and attempted modifying existing methods to suit metal detection
4-Apr	Finished microcontroller programming	Worked on completing Eigenfaces algorithm and testing it- this approach was not successful
11-Apr	Soldered microcontroller, connectors, resistors onto PCB, initial test with button passed	Researched yolo algorithm and its implementation
18-Apr	Soldered power switch, integrated with microcontroller code, and obtained a working model of the project's hardware and main software	Attempted to configure Raspberry Pi to function on basic code, tested it and completed tolerance analysis
25-Apr-22	Final test before demonstration. First PCB fries, proceeded to solder the second PCB	Coded the machine vision algorithm using YOLO

Table 9: Schedule of the main project development

5. Conclusion

5.1 Accomplishments

In this project, we have succeeded in getting nearly all the modules (and thus, subsystems) working. The peripheral module provides expected inputs to the PCB module, including powers (3.27 volt into electronics circuit, and 12.9 volt into power line) along with sensor signals. PCB module works well with the help of microcontroller control module to interpret these signals (high/low signal from sound sensor and translating pulse signal from ultrasonic sensor to distance) and generate control signals for the motors in robot module. Overall, these systems work together to control the motion of the robot car, stopping when obstacles get close (we designed 20 cm as threshold) and entering metal verification mode when sound sensor detects high volume sound from the metal detector (metal found, threshold is adjusted based on the sound of the metal detector). Such final functionality demonstrates the potential to fully implement the robot system for use in metal detection purposes.

5.2 Uncertainties

Along with the design, several unsatisfactory results present. Firstly, due to the limitation of parts and time, the final product has not been optimized in terms of a few parameters: (a) number of cycles (period of time) the robot needs to perform an operation. (b) minimum impact distance, which is the distance required for robot to safely stop without crashing into obstacles. These parameters are essential to both the safety and efficiency of the robot. Given the length of the car (0.5 meter), speed of the car $v = RPS * 2\pi r \approx 80 cm/s$, it requires about 3.75 seconds in total to turn around by a full circle. Therefore, we in the end have set number of cycles to be 15. With each cycle 0.25 seconds, $15 \times 0.25 = 3.75$ secondss are needed to perform a given operation (turn left/right/go straight). But in real time, the speed may be smaller, and some other factors may affect the result, we need physical experiments to confirm the cycle numbers for the robot to properly avoid obstacles while searching an area efficiently.

Secondly, as mentioned in section 2.2.2, we had an unsatisfactory power switch design due to shortage of time. Although overall, it works well in terms of controlling motors to run in single direction, we could use the 4-power-switch design to extend the functions such that the robot car has more well-controlled operations (going backwards, rotate, and so forth). This enables the car to perform more complicated tasks which adapt to various environments during operation (for example, going backwards when encountering obstacles). Some power parameters design may also be improved, including the output voltage when power switch is off. Ideally it should be 0 volt, but our circuit design has a leakage voltage of 0.6 volt. This may either be due to the switch itself, or the non-ideal PCB design (spacing between components is too small which may cause linkage voltage and current). To reduce unnecessary power lost, the voltage near the switch output should ideally be smaller than 0.1 volt.

Another aspect is related to the machine vision module, where when attempting to deploy the same code that has been designed on computer onto the raspberry Pi, we were unable to get the system to function due to the nature of the operating system on board the Raspberry Pi. The files needed to run the code needs a modification of compiler on board which is difficult to do due to backend files used in a darknet. This disables the working model of our machine vision module to be practical. But we believe with some more detailed research into the design of implementation of the darknet program behind the Yolo algorithm as mentioned in section 2.2.4, we would be able to fix the issue and successfully deploy Yolo onto Raspberry Pi.

5.3 Ethical considerations

To illustrate the safety and various aspects important to customers, a few considerations would be highlighted. To begin with, the system is reliant on Lithium-ion batteries, the IEEE code 1725-2021[10] deals with how to properly use such batteries and we have followed the specifications with our design by securing the batteries at the back side the robot car and preventing them from damage and posing risks.

Secondly, after rigorous testing, we conclude that if the product is used in a closed room with a levelled ground, the risk of damaging the robot by regular usage is minimal due to the nature of the wheels we have chosen (front wheels are small backed up by large back wheels). To avoid potential hazards, the battery itself has to be disposed of after 200 cycles of recharge as per the user manual of the manufacturer. In the case that an accidental hazard occurs, the general guidelines outlined in IEEE code involving the safe disposal of rechargeable lithium waste must be followed.

Thirdly, the field of vision of the camera and the data stored are totally accessible to the consumer and we support any additional steps taken by the end user to protect their privacy by allowing the consumer to control modify and or delete the image data stored on the robot which complies with the guidelines outlined in section 7.8.I-1 of IEEE's guidelines outlining the rights of end users.

Lastly, the members of the group are committed to holding the IEEE code of ethics outlined in section 7.8.III to follow the code and hold the standards of being engineers.

5.4 Future work

In summary, given the uncertainties we had during our project design and implementation, 3 potential future works can be done to move our project further to a higher standard.

1. As have discussed, our power switch design in PCB module can be extended to perform a much more comprehensive set of operations using a total of 8 power switches for the 2 motors. Such design can enable the robot car to freely move in forward, backward directions and can also rotate while staying at the same coordinate in space.

2. We only have prototyped a metal detector robot with automation functions. Research has been done before to design robot arms and metal detectors to have more degrees of freedom. In case the robot needs to search a complicated environment, say narrow road, compact construction site, or outer space exploration, we may need to expand the robot arm to move more freely and even change its shape.

3. The machine vision module is still very simple. The robot currently cannot decide which obstacle to avoid, and which to search. As many metal objects are hidden below certain obstacles, we need the robot to be capable of making the correct choice, maximizing safety while keeping the detection more accessible.

Therefore, these are the main future works we can do to advance our project. With a generally working model created, we believe our prototype of the metal detection robotics can indicate a promising future, where such techniques and construction processes can be utilized to create much more advanced automated robots for different purposes and push boundaries of these technologies for a better world for human beings to live in.

References

[1] H. M. Ishak, H. A. Kadir, Lim Chain Fat and Mohd Helmy Abd Wahab, "Autonomous metal detector robot with monitoring system," 2008 International Symposium on Information Technology, 2008, pp. 1-7, doi: 10.1109/ITSIM.2008.4631876

[2] Bin Parsusah, Mujiarto & Sambas, Aceng & Haerudin, Irpan. (2021). Design of Arduino-Based Metal Detector Robot. Solid State Technology. 63. 12401-12411.

[3] Masunaga, Seiji, and Kenzo Nonami. "Controlled Metal Detector Mounted on Mine Detection Robot." *International Journal of Advanced Robotic Systems*, (June 2007). https://doi.org/10.5772/5692.

[4] "Fastener Design Manual," NASA Reference Publication 1228, 1990.

[5] AMS1117-3.3, 1A low dropout voltage regulator, datasheet, Advanced Monolithic Systems, available at:

https://pdf1.alldatasheet.com/datasheet-pdf/view/205691/ADMOS/AMS1117-3.3.html

[6] Raspberry Pi Pico datasheet, 2020 Raspberry Pi (Trading) Ltd, available at:

https://components101.com/sites/default/files/2021-01/Raspberry-Pi-Pico-Microcontroller-Datasheet.pdf

[7] Hollemans, M. (2017, May 20). *Real-time object detection with YOLO*. Machine Think. Retrieved May 2, 2022, from https://machinethink.net/blog/object-detection-with-yolo/

[8] "Firefighter fatalities in the United States," *U.S. Fire Administration*, 07-Apr-2022. [Online]. Available: https://www.usfa.fema.gov/data/statistics/ff_fatality_reports.html. [Accessed: 03-May-2022].

[9] *1725-2011 - IEEE standard for rechargeable batteries for cellular telephones - redline*. IEEE Xplore. (2021, August 23). Retrieved May 3, 2022, from https://ieeexplore.ieee.org/document/6046070

[10] "IEEE code of Ethics," *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 03-May-2022].

[11] M. Turk and A. Pentland, "Eigenfaces for recognition," *MIT directpress*, 18-May-2021. [Online]. Available at: https://www.face-rec.org/algorithms/PCA/jcn.pdf. [Accessed: 03-May-2022].

Appendix A Requirement and Verification Table

Table 10 System Requirements and Verifications—Peripheral module

Part	requirement	Verification Procedure	Results
batteries	$Voltage_{measured}$= $Voltage_{ideal} \pm x$, for $0 \leq x \leq 1$ volt. Tolerance is 1 volt specifically considering the motor's tolerance voltage at 13.5 volt.	Connect batteries to a breadboard or PCB and use multimeter to measure the voltage reading across the PCB system (positive to one connector's positive port, and ground to GND port).	PCB battery (4.8 v) shows 4.79 volt in stable reading. Occasionally measures 5 volts approximately when directly measure the positive and ground leads. Motor battery source (12 v) shows 12.9 volt which is slightly higher than the ideal threshold value but is fine with motor (which has tolerance voltage around 1.5 volt).
Sound sensor	$1.8\ volt \leq V_{digital\ output} \leq 3.3\ volt$ This 1.5 tolerance voltage range is chosen to match microcontroller's high-level voltage.	Connect sound sensor to PCB, or power pin to a 3.3-volt source (and properly grounded), slightly rotate the potentiometer on the sensor clockwise to increase sensitivity until 2 LEDs are turned on, measure the voltage of the digital output pin with a multimeter.	The multimeter reads about 2.7 volt, which is enough to trigger an active high once it is connected to microcontroller
Ultrasonic sensor	Distance sensitivity is at least 20 cm. Such tolerance distance is chosen to ensure the safe operation of the robot car. Future data may support changes to this tolerance.	This is a little tricky to verify because it can only be properly verified once it is connected to either PCB or other microcontrollers/developing board. Once pins are connected, send a 50 microseconds pulse signal in its trigger pin, and collect the distance information with echo pin. Either display the distance measured by using "print" statement in the program or use external connected device to indicate a "stop" operation when the distance measured is below 20 cm	We use the second method to qualitatively show that the distance sensitivity is below 20 cm by stopping the car's motion with hand covering the distance sensor at distance about 20 cm. This is done with PCB and sensor connected. Motors will continuously run unless the distance sensor measures any distance below 20 cm.

Table 11: System Requirements and Verifications—PCB module

Part	requirement	Verification Procedure	Results
voltage regulator	V_{output}= 3.3 $volt \pm x$, for $0 \leq x \leq 0.1$ volt. Tolerance is 0.1 volt because of the sound sensor, which has a lower working voltage at 3.3 volt. Tolerance is pending to change if sound sensor cannot generate enough digital signal voltage.	Like battery verification, a multimeter is used to measure the voltage. But now we need to solder the voltage regulator into PCB firstly and apply PCB power source for measurement (connect positive lead to the V_{out} pin on the regulator directly or touch the power port of one of the connectors; negative lead should be grounded into the PCB).	$V_{output} = 3.27$~$3.29\ volt$ Thus, the voltage is ideal for the PCB system given the 0.01~0.03 voltage fluctuation.
power switch	$V_{output} \geq 0.9V_{in}$ once switch is on, and $V_{output} \leq 1\ volt$ when it is off Tolerance is chosen to be larger than 90% of V_{in} because our power switch can only operate if V_{out} has sufficient voltage (90%)	Solder the component onto PCB firstly. Apply motor power source (or any other testing voltage source) to the input of power switch. Use an external 3.3-volt power source with ground common-grounded into the PCB and positive lead touching the enable pin on the power switch. Measure the switch output voltage with multimeter.	When positive lead touches the enable pin on the power switch $V_{ouput} = 12.8\ volt$~$12.9\ volt$ which is within the minimum input voltage. When enable pin is grounded, $V_{ouput} \approx 0.6\ volt$ This is due to some linkage voltage that will not be enough to drive the motor, therefore is tolerable.
Other signals	$V_{signal} \geq V_{level}$ Or $V_{signal} < V_{level}$ for controlled signals in the module	This is a general debugging verification step, use multimeter to measure output/input pins to see if the signal voltage is within the high/low level voltage limit. Typically, $V_{level} = 1.8$ volt.	For debugging purposes, signals not matched to its appropriate levels will be investigated for short/open circuit cases.

Table 12 High-level Requirements and Verifications

Verification number	requirement	Verification Procedure	Results
1	The modules work together comprehensively. pressing the button push the system to state 1 (motors running); objects to distance sensor < 50 cm, push the system to state 2 (one motor will stop rotating); objects to distance sensor < 20 cm, push the system to state 0 (all motor stops); metal detector alarms push the system to state 3 (motor stops for 40 seconds, waiting for object verification).	Place the constructed robot on a platform where the wheels can freely rotate. Press button, see if the wheels rotate; then cover ultrasonic sensors with hand at 20 cm/50 cm critical distance, see if the motors change their operations based on the requirement. Lastly, use a metal object to trigger sound sensor, see if the machine stops for the correct time interval	The verifications satisfy the requirements.
2	Robot is able to search within a room for a hidden object within 5 minutes	Start the robot at any place. Place a key on the ground with 2 cases (covered and uncovered), then wait for the robot to search for the object.	Not verified. Systems are not entirely integrated. Still lack one power switch, lack of parameters optimized
3	Objects verification accuracy is higher than 80%	Run the robot and manually place objects close to where the robot will move and detect. See if the object is identified for 10 trials with different objects. If at least 8 trials are identified, then the verification is good.	Not verified. Machine vision module has not been thoroughly implemented, thus, will not be tested.