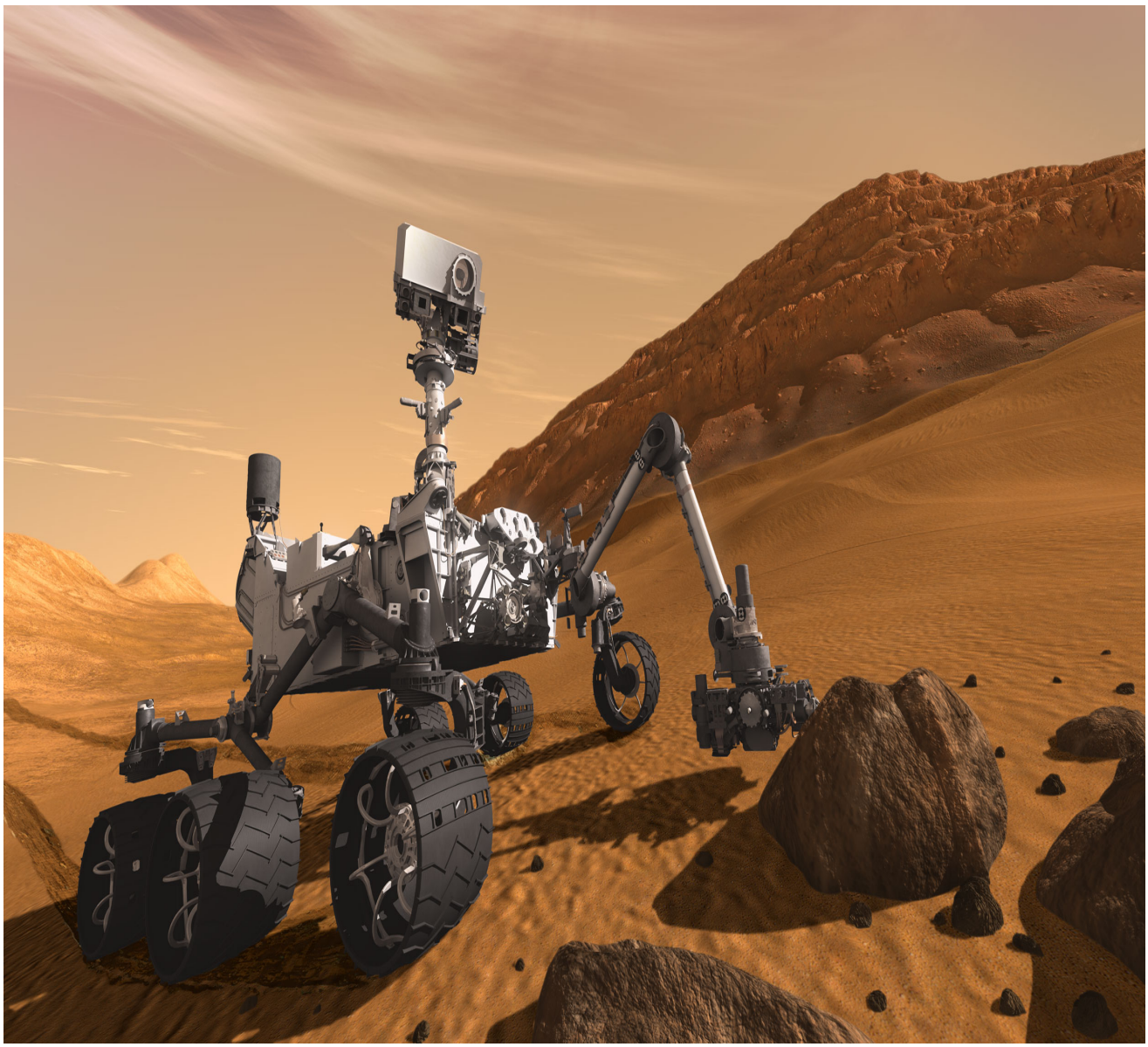


AVC Report

Team Name: Robots on Mars

Student Name: Minping Yang (Rock)
Student ID:300364234

Team Number: Tuesday 14:10-16:00 Team2

Table of Contents

Abstract: 3

1. Introduction: 3

1.1. Motivation: 3

1.2. Scope: 3

1.3 Aims: 3

2. Background: 4

2.1. RPi-Roboshield 4

2.2) Error signals from an image 5

2.3. PID (Proportional Integral Derivative) Control 6

3. Method: 6

3.1 Tools and techniques used 6

3.1.2. Using CAD (computer aided design) to generate 3D structures 7

3.1.3. SSH (Secure Shell) which is applied for controlling the RPI wirelessly 7

3.1.4. Getting Camera Data from the RPi. 7

3.1.5 GitHub which is used for storing and sharing codes online 8

3.1.6 Analog sensors and getting data from them 9

3.2. Process 9

3.2.1 Identify project requirements: 10

3.2.2 Planning: 10

3.2.3 Build robot: 10

3.2.4 Develop Software: 10

3.2.5 Test and Evaluate 11

3.3 The optimal size of wheels and information about battery 11

3.3.1 The optimal size of wheels 11

3.3.2 battery 13

4. Results: 13

4.1 Design parts of the robot 13

4.2 Motors 14

4.4 Values of PID 14

4.5 The sensors 14

4.6 Robot on Mars' timeline of events 15

4.7 Robot testing week results 15

6. Discussion 15

7. Conclusion 18

8. Reference 18

Abstract:

On 19th April 2016, our team, Robots on Mars (ROM) came to a consensus on the top 3 goals that our robot needs to achieve. In this meeting, we as a team discussed about what we have achieved so far, how to achieve the milestones ahead, and how we can continue to work effectively as a team. In addition, we learn how to manage workload by allocating work appropriately to each team member. Robot on Mars did not complete 100% of the maze, just satisfying 65%. The report will show what we have achieved and what we can improve on. Furthermore, this project gives a chance to let us know what challenge we might face and what skills we can contribute to the team in the future of real engineering project.

1. Introduction:

1.1. Motivation:

There are several motivating factors that have motivated the team to work together toward a common goal in this project:

1) Develop communication skills through working effectively as a team
2) Develop coding skills through practical experience with programming and working on different aspects of the vehicle
3) Manage time and money to achieve the project.
4) Develop how to achieve good quality in both software and hardware.

1.2. Scope:

The scope of this challenge was to cover theories that we learnt from other papers besides ENGR101 in this trimester. For example, Raspberry Pis, C programming language, git hub, and 3D printing. All of these technologies were applied for developing the robot. Furthermore, the ENGR101 wiki is the most helpful to solve the problems that we encountered in this project. However, there also are several limitations to this project:

- A budget of 100 virtual dollars (money is created by Arthur Roberts an engineering technician at VUW) is allocated to spend at Parts Bazaar to purchase sensors.
- Limited time as the project has to be completed in 4 main phases with the corresponding deadlines: Project Plan (22nd April), Progress Report (16th May), Robot Test (31th May) and Final Report (13th June).
- The team is made up of 6 team members with background in network and software engineering. The work will be done by the team members without any outsourcing or external support other than from the lecturers and tutors.

1.3 Aims:

There are three main aims for the project:

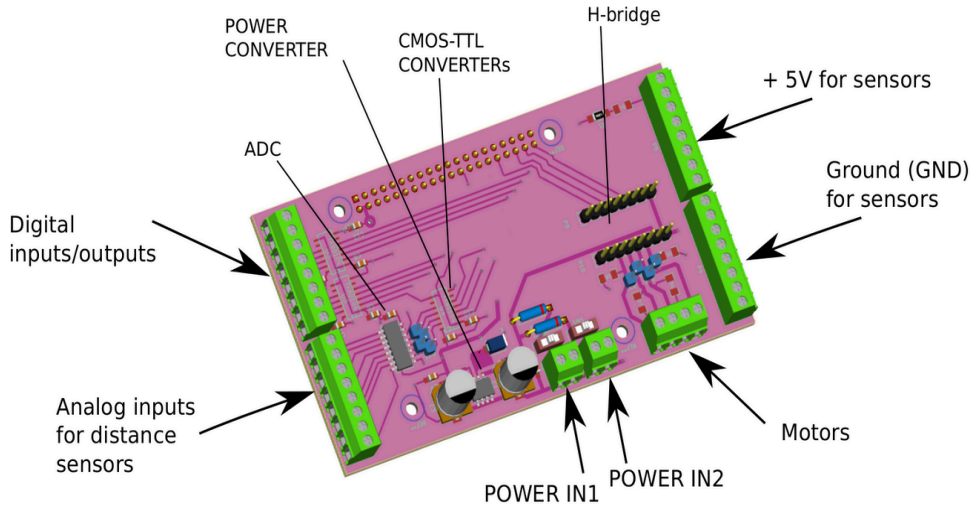
1. Make the robot functional, recyclable, and aesthetic.
2. Make the robot go through the whole maze smoothly.

3. Take the least time for the robot to go through the whole maze.

2. Background:

This project uses RPi-Roboshield, error signals from an image, and PID (Proportional Integral Derivative) Control. This section will give a background to these scientific methods and technologies and show how they have been used to build, program and test the robot.

2.1. RPi-Roboshield



(Eldridge, 2016)

RPi-Roboshield is the most important component for our robot. It is the brain of the robot and is a platform that monitors all activities of the various parts of the robot that are connected to the RPi-Roboshield. However, this robot brain does not function exactly like that of a human being. Therefore, the RPi-Roboshield requires some human inputs through C language programming to adjust the way that it analyses information and perform actions.

RPi-Roboshield provides three main function for the robot: (1) powers the hardware, (2) powers the motors, and (3) . Firstly, it powers other pieces of hardware of the robot that are connected to it, from a battery that is attached to one of its power inputs. There are 2 power inputs (as shown in the figure above with POWER IN1 and POWER IN2) each with a switch. The battery can be connected to either of them. Hardware can be powered by either or both of them.

Secondly, it powers the motors. The two motors are connected to 2 pairs of terminal blocks as shown above. See below for motor control library functions. There are 2 LEDs for each motor, red and green, indicating polarity of the voltage applied to the motors (i.e. direction of rotation).

RPi-Roboshield is a central hub for the sensors that are attached to it. The robot has digital distance sensors that act as its ears and eyes. These sensors are used to detect incoming obstacles and walls that the robot is required to navigate around. As shown by the figure

below, the digital distance sensors are connected to the platform via the “Digital” port, and the power is provided from the battery to the sensors through the “5V” port.

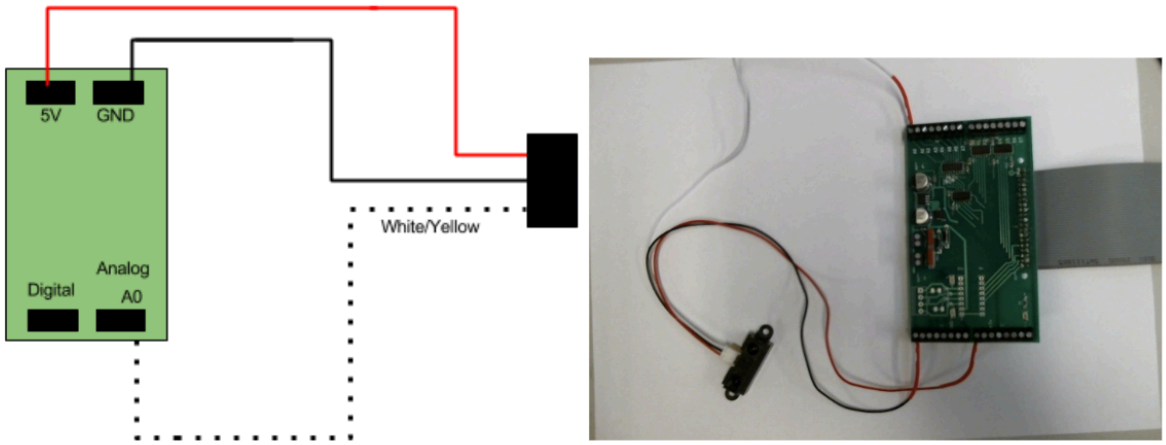
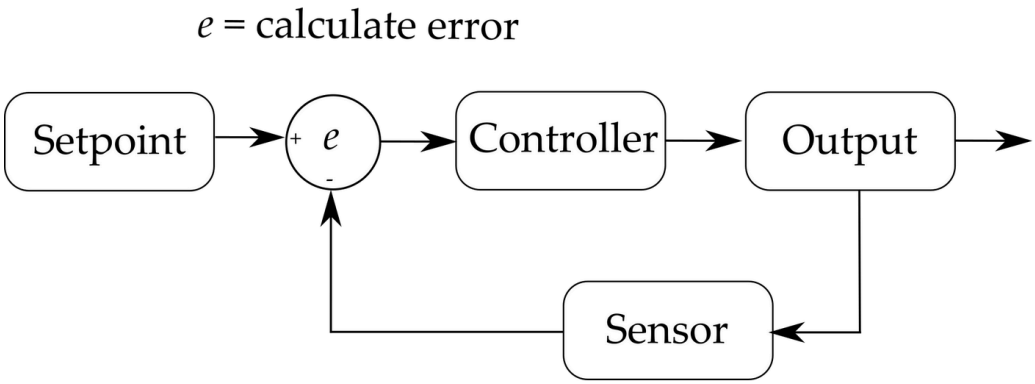


Figure 5: IR sensor connected to RPi analog input pin 0.

(ENGR101 Assignment4)

2.2） Error signals from an image



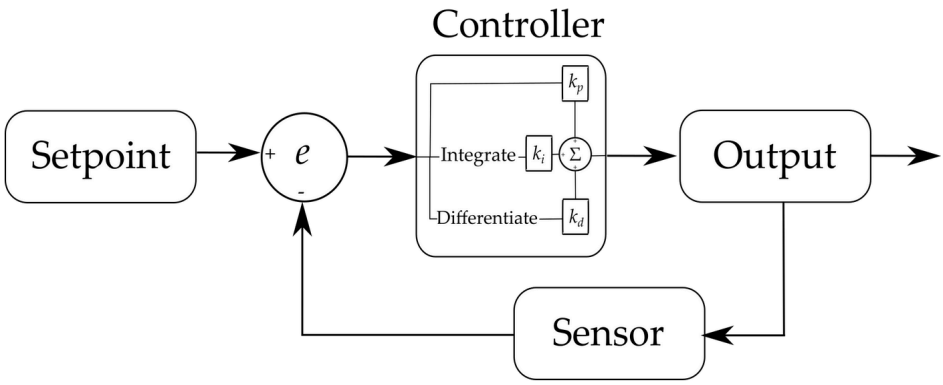
(Eldridge, 2016)

This project uses the error signals technology. Error signals can be thought of as a measurement of how far away a system is from a set point. This requires a target, a sensor, and an element for calculating the error (the difference between the current sensor value and the target value).

The camera receives image data that includes the road ahead for the robot. The codes then analyse a horizontal section of each image in order to distinguish between the white line that the robot is supposed to follow and the black background. The camera being used produces images that are 320 pixels wide by 240 pixels of length. Starting from the left-most pixel p_1, the codes analyse the data and assign a value for each pixel across the entire image section according to its colour. For example, the black background is assigned a value of 1 and the white line is assigned a value of 0. Unfortunately, signals from the pixels will be very noisy because the black background is not perfectly black but rather variation of the same colour. Similarly, the actual colour of the white line is not exactly white but rather a variation of white. The codes use a conditional statement to remove the noise and establish a threshold to decide if a pixel is “white” or “black”. The error signals can then be calculated by the codes as the horizontal distance between the centre of the vehicle and

the centre of the white line, in which the latter is the number 0 in the centre of all the signals. The codes will then coordinate with the motors to direct the vehicle to eliminate the error signals.

2.3. PID (Proportional Integral Derivative) Control



from (Eldridge, 2016)

PID describes a particular way of constructing Controller (in the diagram above) so it responds to changes in the error signal in specific ways. An example of a PID controller is shown in the diagram above.
The strength of each response is tuned by a **constant of proportionality**: e_p, e_i or e_d respectively. This gives the controller the ability to respond to not only the current error value, but the rate of change of the error signal (time derivative of the error) and the net error (time integral of the error) the system has experienced. The strength of each of these behaviours are controlled by three proportionality constants: e_P, e_D and e_I respectively, which can be adjusted to alter the behaviour of the system. The three signals are summed together it give a total output which is used to control the response of the system. (Their relationship with steering angle as the figure shown below)

$$\text{Steering angle} = \overbrace{\text{P}\times e_P}^{Proportional\ term} + \overbrace{D\times e_D}^{Derivative\ term} + \overbrace{I\times e_I}^{Integral\ term}$$

3. Method:

This section looks at the methods used in order to build and program the robot according to the requirements of the project. Section 3.1 highlights the tools and techniques used in developing the robot and short descriptions of the techniques being used to take advantage of these tools. Section 3.2 highlights the process the team went through with this project. Section 3.3 looks at the optimal size of wheels.

3.1 Tools and techniques used

This project uses several major techniques: Using CAD (computer aided design) to generate 3D structures, getting Camera Data from the RPi, SSH (Secure Shell) which is used for wirelessly controlling the RPi, GitHub and analog sensors.

3.1.2. Using CAD (computer aided design) to generate 3D structures

CAD (computer aided design) is used by computer to generate 3D structures. CAD is an invaluable tool for prototyping and, as such, there is a large number of software programs out there to assist with this process including Sketchup, FreeCAD, AutoCAD, Blender and more. FreeCAD is used for developing the parts of this robot since our team prefers 3D manipulation as a way of creation. FreeCAD is a parametric 3D modeler made primarily to design real-life objects of any size. Parametric modeling allows user to easily modify design by going back into model history and changing its parameters.

From (Eldridge, 2016)

3.1.3. SSH (Secure Shell) which is applied for controlling the RPI wirelessly

The Fourth Industrial Revolution is underway and the use of online applications is becoming increasingly widespread throughout the globe. It is therefore essential for our vehicle to be able to be controlled via the Internet. A tool that enables such feature is Secure Shell or SSH. SSH is a cryptographic network protocol for operating network services securely over an unsecured network [referencing]. This network protocol remotely logs user into a computer and gives user access to that computer's command line and control of the vehicle. By using SSH, Raspberry Pi and the vehicle can be controlled remotely through the Internet.

To connect to the Raspberry Pi (RPi) over the network there are three steps that need to be taken:

1) **Obtain IP address of the Raspberry Pi:** on RPi start the command line. Type sudo ifconfig (command reports network parameters of all networks RPi is currently connected to). Locate entry for wlan0 network. Write the IP address down (It can look like 10.140.45.67).

2) **SSH is used for the RPi connect the internet:** On a lab PC start the command line and type ssh pi@10.140.45.67 (the user should use the IP address with one user obtained for his RPi in the previous step). pi is user name for the RPi - it is fixed. If prompted for confirmation type yes. Password for the RPi is also pi - enter it.

3) **Terminating the connection:** User can terminate the connection by pressing Control+C. User are now connected over the SSH link to the RPi and any commands user enters on lab PC will in fact be executed on the RPi.

From (Eldridge, 2016)

3.1.4. Getting Camera Data from the RPi.

The robot needs to have “eyes” to see where it is going. The RPI attached to the robot is equipped and linked with a camera that has a 55-degree angle view, which comes along with a camera mount that allows easy angle adjustment. In addition, the camera has a frame rate of 80 frames per second. Images from the camera are processed by the codes written in C programming language to guide the robot through the testing ground. It is noted that users should choose an appropriate height and angle of the camera carefully so that the robot is able to see where it is going. The choice of height and angle of the camera may require trials and errors in order to find the optimum camera position and angle.

After a process of trials and errors, the optimum angle for the camera is set facing the ground at right angle so that the vehicle will be able to detect the white lines on the test ground. The camera is set at approximately 10cm above the ground to enable to the vehicle to distinguish between the white lines from the other colour.

A quick way to check that camera is functioning is to open Command Line Interface and type raspivid -o. If camera is working the image will be displayed on the computer screen for a short period of time.

There are a few methods in libE101.so which are used for Getting Camera Data from the RPi.
take_picture()
int take_picture() - reads current image from the camera. Stores the image in memory.
get_pixel()
char get_pixel(int row,int col,int color) - returns colour value for the pixel. If **colour** equals 0 then red component is returned; 1 - green component; 2 - blue component. Value of colour components are char type (0-255). If **colour** equals 3 then "whiteness" (proper term is luminosity) of the pixel is returned: $(red+green+blue)/3$. You can recall that when all three colour components equal 0 then pixel is perfect black. If all components equal 255 the pixel is perfect white.
set_pixel()
char set_pixel(int row, int col,char red,char green, char blue) - sets pixel to the specified colour (red,green,blue) in the image. Once function is applied to same pixel subsequent calls to to ***get_pixel()*** will return values you set with ***set_pixel()*** - not original pixel colours.
display_picture()
int display_picture(int delay_sec,int delay_usec) - displays still picture on the screen for specified time. Latest image obtained with ***get_picture()*** will be displayed.
save_picture()
int save_picture(char filename[5]) - saves picture as bitmap (ppm format) into the file with name specified; **.ppm** is added to file name.
open_screen_stream()
int open_screen_stream() - opens area on the screen in which video output of the camera will be directed.
update_screen()
int update_screen() - updates video output area of the screen. Uses last picture taken. You still should use ***int take_picture()*** to obtain the picture.
close_screen_stream()
int close_screen_stream() - stops directing camera output to the area of the screen. Does not re-draw the screen area.

from (Eldridge, 2016)

3.1.5 GitHub which is used for storing and sharing codes online

GitHub is a web-based Git repository hosting service. It offers all of the distributed revision control and source code management functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

This section will show several functions of GitHub which is used for this project in details.

- **Storing code on GitHub**: create a code repository (it is an online folder for all the code and documentation relevant to a particular project). Now that navigate to user's profile (https://github.com/username), select the 'Repositories' tab and then select the green 'New' button. Give repository the name 'ENGR101-2016' and the description C code for controlling Raspberry Pi 2s in ENGR101 during 2016. Choose 'Public' and check the Initialize this repository with a README box. A README is a short text document that introduces the rest of the files and code within a repository. It will often include installation and usage instructions, as well as a To DO list of what needs to be fixed or improved on in the repository. Finally, select the green 'Create Repository' button. Now it's ready to start uploading code to the repository. If users want to copy and paste code, just choose the `New file' button, name file and then put code in there. Saving file with the appropriate extension (e.g. as a '.c' file for C code or a '.java' file for java code) will enable syntax-based highlighting on GitHub for code. A better way of uploading multiple files however is to use the 'Upload files' button where users can simply drag and drop existing files and upload them.

- **Clone a repository** (or just a piece of code) from the GitHub site to whatever computer we're currently working on so that we can execute the code. This is what the git clone command does. Open the command line (black square on the bottom left of your desktop) and type the following command (or copy and paste it): git clone https://github.com/RobotsOnMars/Chappie.git then hit enter. git clone tells the computer to use git to copy the entire repository. The final url says where the repository is located (in this case in RobotOnMars's GitHub repository).

- **Monitoring Code Changes:** With code displayed on GitHub, user can choose the 'History' button and see what edits were made when and the 'Blame' button to see specifically who made those edits. Note specifically that the History button also allows user to get a copy of code before each new commit was made, which essentially allows user to roll code back in time. With the history view selected examine one of the previous commits to file (by clicking the string of numbers and letter left of the '<>' button). Now user can see the full changes to the file colour-coded for convenience and user can event common on the changes that were made.

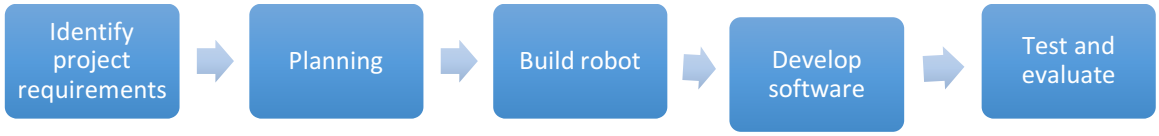
- **Repository Activity:** As GitHub is often used to manage projects around code as well as the code itself. To see important statistics for repository, from main repository directory, select the 'Graphs' tab which will show a visualization of who has contributed to team code and when.

Form (Eldridge, 2016)

3.1.6 Analog sensors and getting data from them

`int read_analog(int ch_adc)` - reads analog voltage supplied to channel **ch_adc**, between 0V and 5V. An output of 0 corresponds to 0V, an output of 1023 corresponds to 5V and an output of 512 corresponds to 2.5V. There are 8 analog channels/inputs labelled on the PCB with **A0, A1, ... A7**.

3.2. Process



3.2.1 Identify project requirements:

In the first team meeting, Robot on Mars make several milestones in 6-7 weeks according to 4 main deadlines. During the first team meeting, team members were allocated 4 main roles: Minping Yang and Brandon Muller (software development), John Grant (team leader), Ryan and Jack (hardware development) and Zac (operational test). According to the project requirement, divide the maze into two big parts, for the first part, find the way to follow the white line by getting data from camera. Thus, in the first part, the robot does not have to install sensors. For the second part, Use sensor to get the data about the distance from the walls to the car. In addition, condition statement which make the robot go to the right direction in the maze was discussed as well during the first meeting. In the first meeting, the team talked about future meetings expectations.

3.2.2 Planning:

By discussing in the first team meeting, the project plan is written according to the content of the discussion. The team communicate channel in slack (https://robots-on-mars.slack.com/messages/general/) and team repository in GitHub are created, which is good way to share code and researches together and communicate with each other instantly. Weekly blog and Weekly checklists also are created to manage time and allocate appropriate workload for each team member by recording weekly project process.

3.2.3 Build robot:

Robot on Mars built the robot by connecting camera, sensors, motors to RPi. At the same time, use five small pillar to support between top board RPi and bottom board RPi. The wheels were installed in underpan of the vehicle to make the robot run. There is only one back wheels used to follow two front wheels running.

3.2.4 Develop Software:

Firstly, the maze test is divided into two big parts for designing codes –line maze and walled maze parts, which resulting that a Boolean variable(in_line_maze) (Boolean is a type of variable that returns either true or false) was set, then it is put into a condition in if-else statement. if(in_line_maze){} else{}

Then, in order to simply find the white line in front of vehicle and get valid camera data, colour data was used for only one horizontal line in the camera view.

void get_line(int *line_buffer, int offset, int start, int end, int is_vertical);

This method can record the colour data into array- line_buffer .

Before using void get_black_white(int *line_buffer, int *input, int length

The colour value in the array displayed like below:

222	200	233	10	20	50	140	170	200	223

After using void get_black_white(int *line_buffer, int *input, int length

The colour value in the array displayed like below:

1	1	1	0	0	0	1	1	1	1

However, the robot “eyes” only should judge the colour either white or black. In fact, it is impossible that that white line is 100% white and black ground is 100% black. In this case, THRESHOLD is defined to 100(By testing, the best value for threshold is 100), which is for divide 0-255 colour value regarding into either white or black. For this reason, void get_black_white(int *line_buffer, int *input, int length) – the method is created to convert colour values into either 1 or 0(0 represents white,1represents no white in this project).

3.2.5 Test and Evaluate

Some hardware problems were found and solved by testing, for example, camera function was tested by extern "C" int take_picture();. By observing the image, the camera should read in the images that are upright. In the beginning, the team tried to use the codes to match the images. However, it was easily to make mistakes for using the incorrect orientation of the image. To solve the problem, the camera was re-installed by rotating by 180 degrees.

Some software problems also were found and modified by testing. In the beginning, vectors are decided to use for storing colour data instead of using array. Additionally, vectors can only be used in C++ programming language but not C language. However, the vehicle didn’t store colour data when the vehicle was running in the maze. Under the suggestion from Arthur Roberts, Robot On Mars decided to use array in C programming language instead of using vectors. C++ is quite difficult being used correctly by first year students. In addition, by testing, the PID control value also is confirmed. In the beginning, A general PID tuning approach was tried:

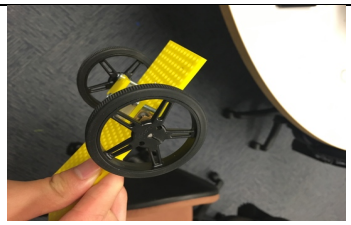
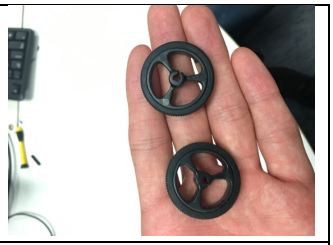
- Implement PID
- Set both e_i and $e_d = 0$.
- Then increase e_p until system begins oscillating.
- Then increase e_i until oscillation stops.
- Then increase e_d until loop reaches target in acceptable time.

(ENGR101 Lecture note 24)

By following this method, Robot On Mars find e_p=0.7 e_i=0.1 e_d=0.2. which make PID controller work well.

3.3 The optimal size of wheels and information about battery.

3.3.1 The optimal size of wheels

			
Size of wheels(mm)	40	30	15

As the above table shown, three pairs of different sizes of wheels are compared.
By doing researches and experiments, Robot on Mars decided to use the medium size of the wheels (30mm).

In running straight line situation,

$$V(\text{velocity}) = \omega(\text{angular frequency}) \times r(\text{radius}) \quad \text{--------equation1}$$

$$\omega(\text{angular frequency}) = \frac{2\pi}{T(\text{period})} \quad \text{--------equation2}$$

$$n(\text{rotation rate}) = \frac{1}{T(\text{period})} \quad \text{--------equation3}$$

Plug equation3 into equation2

⇨ $\omega(\text{angular frequency}) = 2\pi \times n(\text{rotation rate})$ --------equation4

Plug equation4 into equation1

⇨ $V(\text{velocity}) = 2\pi \times n(\text{rotation rate}) \times r(\text{radius})$ --------equation5

As the supply power determines (n) the rotation rate, and the motors have same rotation rate since the motors which are supplied same battery in this project. According to equation 5, the biggest size of the wheels has the highest speed.

However, the fastest one is not absolutely the most optimal one since flexibility is also an important element to affect the quality of the wheels. To compare the wheel's flexibility, Robot on Mars made 3 times tests. The time the robot takes when the robot which was equipped with different sizes of wheels run in the same curvy pattern was recorded. (As shown in Table 3.1 below)

Table 3.1 Time taken for the robot to complete a curve (seconds)

Size of wheels (mm) / Trails	40	30	15
First time	122	118	135
Second time	121	107	138
Third time	118	113	130
Average (s)	120.3	112.6	134.33

3.3.2 battery

6.6 voltage receiver and 1700 mAh 20c discharge.
This is the battery used to power the robot.
It performed to expectation.

4. Results:

The Results will be structured in the following format, and all researches and code had been already uploaded in Robot On Mars' GitHub (https://github.com/RobotsOnMars), at first all parts and features which were used for meeting this project will be shown. Then, the coding for quadrants was done will be displayed and explained in details. Finally, all the parts and features which were specific to the quadrant will be shown and to finish there will be a brief result given on the team.

4.1 Design parts of the robot

In terms of recyclability, the robot was built to a high standard. The robot was covered by ice-cream sticks and all parts were either screwed or cable-tied to the structure, resulting in a high level of recyclability. (as figure4.1 shown below) In addition, the really simple parts are created by using 3D printer. (as figure 4.2 shown below). The two longest parts is for holding the ice-cream stick in the both sides of vehicle. The shortest part is for holding two sensors in 45 degrees. The last part is to avoid battery slipping out of the vehicle.

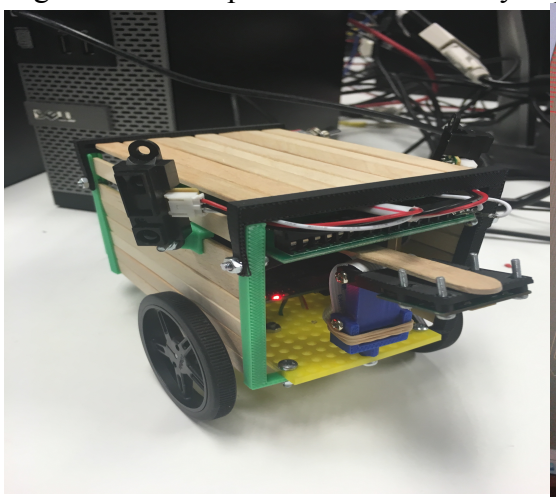


figure 4.1

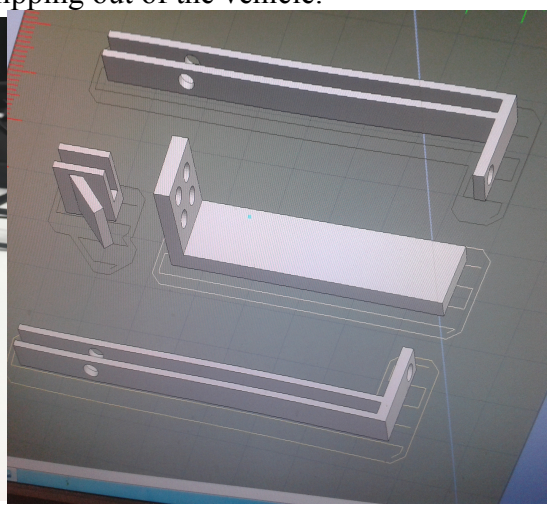
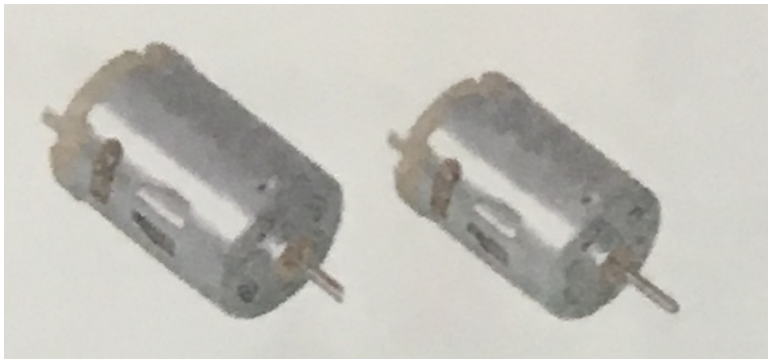


figure 4.2

4.2 Motors



There are two motors used for running front of wheels
Code which operate for the motors:

```
extern "C" int set_motor(int motor, int speed);   //import the library for controlling the motors
void move(double direction, double speed) {    //method for move the vehicle
    double left;
    double right;
    if (direction > 0) {                          // if direction >0, which represents the vehicle
        left = 1;                                 //turn right by making left wheels speed greater
        right = 1 - 2 * direction;                //than right wheels',
    }
    else {                                        //else, turn left by making right wheels speed
                                                  //greater than left wheels',
        right = 1;
        left = 1 + 2 * direction;
    }
printf("%d %d", (int) ((255 * left * speed * LEFT_GAIN)), int (255 * right * speed *
RIGHT_GAIN));                                     //print out the left wheel speed and right wheel
/                                                  //speed, which is continent for testing
    set_motor(1, (int) (255 * left * speed * LEFT_GAIN));         //motor 1 is left wheel
    set_motor(2, (int) (255 * right * speed * RIGHT_GAIN));     //motor 2 is right wheel
}
```

4.4 Values of PID

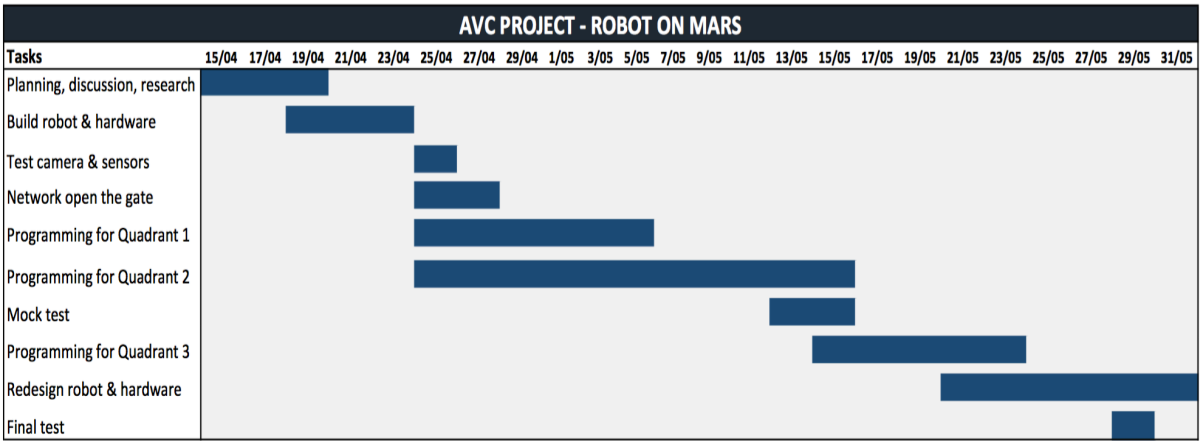
By a number of tests for this robot, Robot on Mars find the most optimal values for PID controller three parameters (Proportional, Integral and Derivative).

```
 #define P_GAIN 0.7
#define I_GAIN 0.0
#define D_GAIN 0.0
#define P_MAX 0.6
#define I_MAX 0.0
#define D_MAX 0.0
```

4.5 The sensors

By some researches done by the team members, our team decided to use the Medium Range IR sensor (analog). Because the short range could only detect things right in front of the vehicle, the medium range was more reliable for detecting objects. Additionally, the short range IR sensors (digital) only have on or off values, rather than the analog one sensors being able to return multiple values. Therefore, we chose the Medium Range IR sensor (analog).

4.6 Robot on Mars' timeline of events



4.7 Robot testing week results

Robot on Mars(ROM)' vehicle went through 65% of the maze in under 15 minutes on 31st May. In fact, on 15th May, the vehicle was able to go through 65% of the maze. ROM spent half a month in making the vehicle going through quadrant 3. During the test, the robot was able to go through quadrant 3 from quadrant 2 of the maze, but the robot could not do the quadrant 1 and quadrant2 because of some unknown programming errors. By testing and debugging, unfortunately, the robot became unable to function properly. Robot on Mars have to recover the code back to the code which was written on 15th May.

6. Discussion

Overall, Robot On Mars did good for this teamwork, even though the robot didn't go through the all quadrants of the maze. Robot on Mars experience more meaningful and important thing is how to work well with team members, what Robot on Mars learnt from mistakes in both software and hardware way, and how to manage time and money for a project. For example, Robot on Mars (for the first couple of weeks) only worked on the robot during the weekly two hour laboratories, and too much time is used for discussing not trying and testing.
As a result, there are lots of code is typed first and it might work well in theory. In addition, the unsuitable programing language is decided to use, which waste lots of time and energy in a wrong way. Under the suggestions were given from Elf, Robot on Mars should not make things too hard. Just use the basic information form lecture and the wiki which were provided by Elf. Because, in the beginning, vector was used for storing the colour data from camera instead of array. However, most of team members did not know how to use it and how to use C++. Hence, Robot on Mars has to try hard to write code in C++ programming.

Robot On Mars as a team collaborated effectively even though the robot did not complete all quadrants of the maze. Another aspect that is relevant to limited team time was access to the

co145 engineering lab. Although in the weeks leading up to the challenge the team had mostly unrestricted access (except for when another ENGR101 lab was running). The laboratory was always packed and it was impossible to get a seat when it came to the week leading up to challenge the laboratory. This did not allow team to collaborate properly on the challenge. Next year, I would recommend having a backup lab with another practice track so that groups can work together and are not turn away from the lab. During the week of challenges, the team was not allowed to use the track during other groups' lab times. As a result of this, Robot on Mar was forced to make a practice test course. (see figure6.1 below) We could not accurately test robot on this course either as the reflectivity of the mats were slightly different to that of the proper course.

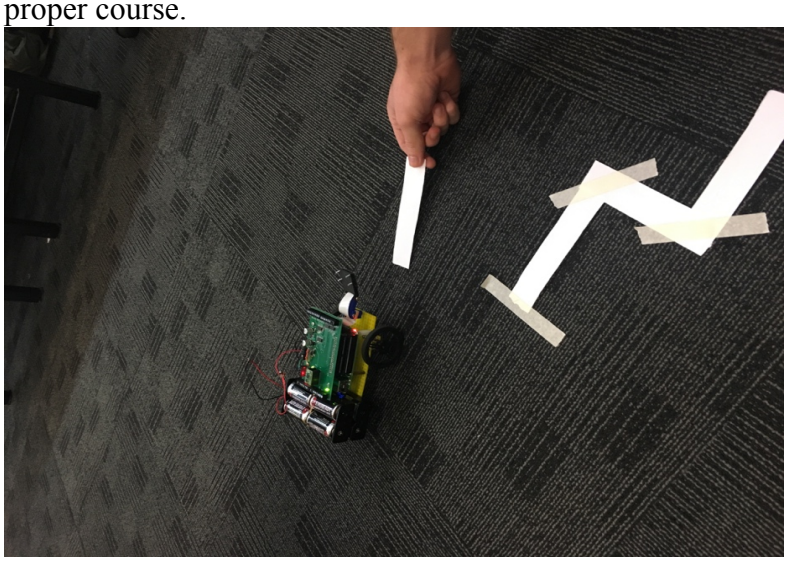


Figure 6.1: practice courses with quandrant1 and quandrant2

We had to deal with numerous hardware problems, in particular a lot of the connections between the camera-holder and the back wheels. The camera worked well in getting the images, but it was becoming difficult for us to get the camera to be fixed to its position facing the ground at a direct angle. This created uncertainties around the accuracy of the data received from the camera and therefore the effectiveness of our codes.

Different threshold, PID, speed values will refer to different angles. As a result, at certain times, the robot did not always work well using the same codes. By discussion, Robot on Mars try to use 3D printer print a part for fixing the camera, however, Robot on Mars did not regard this as a crucial problem. An ice-cream stick was putting into the robot to fix the camera (figure 6.2), which result in the angle between camera and the ground not being fixed. This indirectly affect every test when the camera was touch by accident. In terms of the back wheel, they negatively affected how the robot was turning. The initial design of the back wheel was not flexible for moving left and right. It misled the robot to run toward the wrong direction. In addition, the back wheel' height is lower than the front wheels. As a result, the robot is unbalanced. Therefore, the battery holder also was designed in order to avoid the battery to fall out of the vehicle. (shown in figure 6.3)

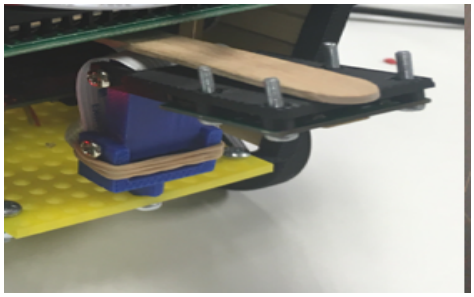


Figure 6.2 Ice-cream stick for fixing camera

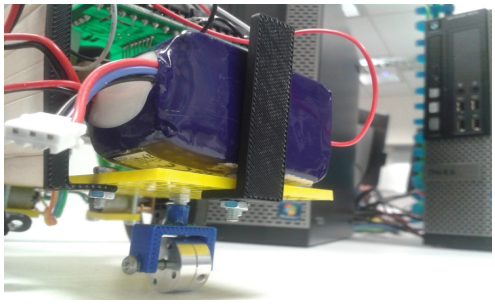


Figure 6.3 Back wheel and battery holder

There were a number of software related issues that came up throughout this project. It was however the case that whether a problem was related to software or hardware was difficult to be analysed only according to observation from the test. For example, Robot on Mar spent 1 week to program the code for Quadrant 3. We found that it was very hard to merge the codes which were written by different programmers into a whole to run. Originally, the robot was able to complete Quadrant 1 and 2. However, due to the new codes added and merged, the robot could not even follow the white line. The team attempted to test and debug some potential errors. Unfortunately, the team could not find the appropriate solution to merge the Quadrant 3 codes. As a result, the team recovered the codes to 2 weeks ago. This experience let every member of the team down because we did not have the time to move forward before the big robot test day. As we have all learned from our experience, we feel that we have spent too much time to focus on creative and revolutionary ideas instead of focusing on what the project actually requires us to do and accomplish.

The 6.6V battery, in which the robot was using to power its system caused a lot of problems with programming and making the software robust. This was mainly due to the fact that the batteries ran out quickly as they were being used up by the robot. This meant that a brand new fully charged battery would be performing at below half its power in 2-3 test runs with the robot. This caused issues because it changed the time taken for the robot to do things e.g. turn 180 degrees, as the motor speeds and time taken where constantly varying. In addition, there was no way of identifying batteries and groups were not assigned batteries so every time as new battery was taken it power given by the battery was changing values to allow for error correction caused by the batteries and we tested this in the maze to insure this was true. However, a group the performance of the robot varied even after the systems we had in place which meant in the odd run the robot would turn to far or may be a little slower than other runs.

In terms of teamwork, we worked very well together as a team with effective collaboration and open communication. However, the level of contribution varied across team members throughout the course of this project. From a point of view, this can be attributed to uneven allocation of workload that we set out at the very beginning of the project. As a result, some team members were tasked with substantially more workload than some other team members. The scope of this project also limited our ability to share our workload because during lab sessions we had only one robot for testing. Overall, working as a team on this project gave the team members great insight into how to work effectively as a team and how to manage time and workload efficiently.

7. Conclusion

In conclusion, the aim of this project was to successfully design, construct and program an autonomous vehicle which could navigate a white line in a maze. The autonomous challenge allowed us to think critically and constructively in a group environment under cost and time constraints, it introduced us to what is now commonplace amongst industry and how most tasks these days are not done independently. I have learned more about C programming by having the opportunity to apply what I have learned in class to hands-on practical experience. On the other hand, the communication skills of the team members were also improved. In addition, for this project we will have the fundamental knowledge of completing a task under budget, in time, a steep learning curve and successfully as a group, as these skills are vital for the IT industry and throughout our university life.

8. Reference

1. Eldridge, E. (2016, April 12). RPI-Roboshield. Retrieved from Github: https://github.com/kaiwhata/ENGR101-2016/wiki/RPi-Roboshield

2. Eldridge, E. (2016, May 5). PID (Proportional Integral Derivative) Control. Retrieved from Github: https://github.com/kaiwhata/ENGR101-2016/wiki/PID-%28Proportional-Integral-Derivative%29-Control

3. Eldridge, E. (2016, May 5). Error signal from an image. Retrieved from Github: https://github.com/kaiwhata/ENGR101-2016/wiki/Error-signal-from-an-image

4. Eldridge, E. (2016, April 6). Git usage. Retrieved from Github: https://github.com/kaiwhata/ENGR101-2016/wiki/Git-usage

5. Eldridge, E. (2016, May 6). CAD (Computer Aided Design). Retrieved from Github: https://github.com/kaiwhata/ENGR101-2016/wiki/CAD-%28Computer-Aided-Design%29

6. Eldridge, E. (2016, April 6). Camera. Retrieved from Github: https://github.com/kaiwhata/ENGR101-2016/wiki/Camera

7. Eldridge, E. (2016, April 5). Wireless control of the RPi (SSH). Retrieved from Github: https://github.com/kaiwhata/ENGR101-2016/wiki/Wireless-control-of-the-RPi-%28SSH%29

8. Eldridge, E. (2016, May 6). ENGR101 lecture note retrieved from: https://ecs.victoria.ac.nz/foswiki/pub/Courses/ENGR101_2016T1/LectureSchedule/ENGR101_Lecture24.pdf