Michael Keogh

Nicola Watt

Thuto Luphahla

Milestone 4

Group 12

**Final Report**

Contents

[1 Introduction 2](#_Toc148557608)

[2 System level Design 3](#_Toc148557609)

[2.1 Design Overview 3](#_Toc148557610)

[2.2 Submodules 4](#_Toc148557624)

[2.2.1 Power and Hardware 4](#_Toc148557625)

[2.2.2 Sensing 6](#_Toc148557626)

[2.2.3 Arduino 7](#_Toc148557627)

[2.3 Interfacing 8](#_Toc148557628)

[2.4 Acceptable Test Procedure Summary 9](#_Toc148557629)

[2.5 Design Concepts and Justification 10](#_Toc148557630)

[3 Electrical Design 13](#_Toc148557631)

[3.1 Power Calculations 13](#_Toc148557632)

[3.2 Microcontroller Resource Allocation 14](#_Toc148557633)

[3.3 Power Design and Safety 16](#_Toc148557634)

[3.3.1 Power Design 16](#_Toc148557635)

[3.3.2 Power Safety 17](#_Toc148557637)

[3.4 Schematic 19](#_Toc148557638)

[4 Motion Control 20](#_Toc148557639)

[5 Line Sensing 20](#_Toc148557640)

[6 Treasure Maze Solving 20](#_Toc148557641)

[7 Conclusion 20](#_Toc148557642)

[8 References 20](#_Toc148557643)

# 1 Introduction

This report details the high-level design and execution of an autonomous maze-solving robot. The primary goal was to develop a robot capable of autonomously navigating through a maze, tracking lines, identifying objects, and determining the closets object to a designated detect line. An overview of the design concepts and justification for choices is discussed. The electrical design of the robot is highlighted, including power supply calculations, resource allocations and a schematic of the PCB layout. Finally, the functionality and algorithms implemented for the robot are included, namely for motion control, treasure maze solving and lining sensing.

The system level design is an overview of design concepts and justification for choices. The robot can be simply modelled into a block diagram to provide a rough view of the interfacing between components. The model can be split into three sub-modules: hardware and power, the Arduino and sensing. Analysing the submodule’s requirements and specifications in conjunction with how these systems interact with each other allows for the best design choices to be made.

The electrical design details power calculations, power supply design/safety and schematics. Including safety features implemented in the PCB power connections. Furthermore, the microcontroller resource allocation is presented, indicating how optimal functionality from the Arduino Nano33 IoT was achieved. A design of the entire Veroboard circuit is further provided and explained.

The primary functions of the robot are motion control and line sensing. For each of these actions, the pseudocode and flow diagrams (i.e., state flow from Simulink) are included. The effectiveness of these algorithms is analysed, and the physical results are discussed.

The treasure maze solving algorithm design speaks to its implementation and. The modular design of software down to function calls and including interrupt structure are included. Additionally, the pseudocode and flow diagrams (i.e. state flow from Simulink) are presented.

# 2 System level Design

## 2.1 Design Overview

Key:

Power

Logic

GND

Physical

BATTERY

# 

7.4V

7.4V

# 

Veroboard

# 

3.3V

LLC

# 

5.0V

5.0V

5.0V

3.3V

Motor Driver

ARDUINO

# 

3.3V

# 

5.0V

3.3V

5.0V

LLC

# 

5.0V

3.3V

ULTRASONIC

SENSOR

# 

LINE SENSOR

LINE SENSOR

LINE SENSOR

LINE SENSOR

# 

# 

ENCODER

(L)

ENCODER

(R)

# 

# 

# 

MOTOR (R)

MOTOR (L)

Ground Plane

*Figure 1: Block Diagram of Submodules*

## 2.2 Submodules

The three modules can be analysed by looking at their respective requirements and specifications. Using these, the best design choice was determined and implemented.

### 2.2.1 Power and Hardware

#### Requirements

**Hardware Structure:**

The entire electronics must occupy a Veroboard of at most 10cm x 5cm.

No breadboard usage.

The use of appropriate hardware is provided. Namely, sensors, 3V7 batteries, logic level converters, motor driver, 2WD mobile robot platform and motor encoders.

The 2WD mobile platform for the Arduino is required to house all circuitry.

Hardware must be assembled efficiently and effectively. Axle length must equal 13.6cm, with a wheel diameter of 6.2cm.

No more than five-line tracking sensors can be utilized.

**Power Structure:**

The entire circuitry must operate on the required voltage (2 x 18650 batteries (3.7V each) in series minimum and maximum voltages) and from the bench PSU.

#### Specifications

**Arduino:**

The microcontroller on the Arduino Nano 33 IoT runs at 3.3V. The device cannot be supplied with more than 3V3 to its Digital and Analog pins (5V pin permanently unconnected.) The nominal input voltage is 5V to 18V. The digital IO levels are rated at 3V3. The DC current per input/output pin is 7 mA.

**Ultrasonic Sensor:**

The ultrasonic sensor must be connected as follows: powered using a Vcc pin, grounded at the GND pin, whilst the Arduino is used to send impulse signals to TRIG.

The ultrasonic sensor has the following power and hardware specifications: receives 5V at the input. The output voltage specified is 3V3.

**Logic Level Converters:**

The bi-directional logic level converter has the following hardware and power specifications: must step down 5V signals to 3V3, and further step up 3V3 to 5V at the same time. This accounts for different subsystem voltage specifications. High voltages must connect to the HV pin, low voltages must power the LV pin and be ground at GND.” The device has 4 channels and is compatible with 1V8, 3V3 and 5V.

**Dual H-bridge Driver:**

The dual H-bridge driver is specified at 5V. The drive voltage ranges from 5V to 35V. The Vs voltage must be >= 7V for the 78M05 to regulate correctly. The motor power supply must be connected to the battery supply and GND, through which the Arduino connects using GND

**2WD Mobile Platform:**

- Gear Ratio 1:120

- No-load speed (3V): 100RPM

- No-load speed (6V): 200RPM

- No-load current (3V): 60mA

- No-load current (6V): 71mA

- Stall current (3V): 260mA

- Stall current (6V): 470mA

- Torque (3V): 1.2Kgcm

- Torque (6V): 1.92Kgcm

- Size: 55mm x 48.3mm x 23mm

- Weight: 45g

**Line Tracking Sensor:**

The line tracking sensor for an Arduino has the following power and hardware specifications: a power supply of 3V3 to 5V, a weight of 10g and a module size of 10mm x 28mm.

**TT Motor Encoders:**

The TT motor encoders have the following power and hardware specifications: voltage signal is +5V, current <20mA and a weight of 20g.

### 2.2.2 Sensing

#### Requirements

**Line Tracking Sensor:**

Compatible microcontroller board such as the Arduino Nano 33 provided.

Appropriate wires and connectors to connect to the Arduino board.

Line-following algorithm.

A visible line to follow.

**Ultrasonic Sensor:**

Compatible microcontroller board such as the Arduino Nano 33 provided.

Appropriate wires and connectors to connect to the Arduino board.

An obstacle detection algorithm.

Properly mounted position on the robot.

A step-down converter.

#### Specifications

**Line Tracking Sensor:**

A detection range of 1-2 cm from the surface the robot moves on.

Line-following algorithm to process the sensor’s data and make decisions about the robot’s movements based on the detected line.

A visible line to follow, contrasting the surface that the robot moves on.

**Ultrasonic Sensor:**

An obstacle detection algorithm that can detect how far away an obstacle is from the robot anywhere within a distance range of 2-400cm.

Properly mounted position on the robot to ensure it can detect obstacles within its narrow beam of 15 degrees.

A logic level converter to output a 3.3V signal to the Arduino.

### 2.2.3 Arduino

#### Requirements

**Switch:**

A signal from the switch must trigger the Arduino to start its maze-solving algorithm.

**Line Tracking and Following:**

The Arduino must process data received from the line tracking sensor in real-time to meet the line following requirement.

The maze-solving algorithm must ensure that the robot explores each ‘branch’ of the maze to guarantee that all objects are found.

**Motor Control:**

The Arduino must control the motor driver to alter the direction of the robot as necessary.

The Arduino must trigger the motor to stop upon reaching a detect line and start again once the distance to the object has been measured

**Object Detection:**

The Arduino must process data received from the Ultrasonic Sensor to determine the distance of an object from the detect line.

**Object Sorting:**

The robot must be able to identify the object which is closest to the detection line.

It must also be able to ‘remember’ where this object is and then return to it.

#### Specifications

**Input Power:**

The Arduino has a nominal input voltage of between 5-18V which it will receive directly from the batteries (i.e. 7.4V).

**I/O Power:**

The analogue and digital pins of the Arduino operate at a hard limit of 3.3V.

**Memory:**

The Arduino Nano 33 IoT has 256 KB of SRAM with 1MB flash.

**Physical Specifications:**

The Arduino board is 18 mm by 45 mm in size.

The Arduino has a mass of 5 grams.

**Code:**

MATLAB 2023b and Simulink will be used to programme the Arduino with its maze-solving algorithm.

## 2.3 Interfacing

**Batteries – Arduino**

The Arduino receives power directly from the Batteries which supply it with 7.4V.

**Batteries – Motor Driver**

The Batteries also directly supply 7.4V of power to the Motor Driver.

**Arduino – Motor Driver**

Instructions are sent from the Arduino to the Motor Driver via a logic level converter which steps the voltage up from 3.3V to 5V.

**Arduino – Ultrasonic Sensor**

The Ultrasonic Sensor is powered via the H-Bridge 5V line.

Data is sent and received between the Arduino and the Ultrasonic Sensor via a two-way logic level converter which steps the voltage up and down between 3.3V (for the Arduino) and 5V (for the Ultrasonic Sensor)

**Arduino – Line Sensors**

The Line Sensors are powered by the Arduino’s 3.3V line.

Data may be directly transferred from the Line Sensors to the Arduino since both operate at 3.3V.

## 2.4 Acceptable Test Procedure Summary

Table 1: Test Procedure Summary

| **Test Aspect** | **Description** |
| --- | --- |
| Power Circuitry | Ensure the entire circuit operates at required voltages using bench PSU and multimeters. |
| Motor Driver Operation | Confirm the motor driver's ability to control speed and direction on mazes of varying sizes and complexity. |
| Switch-to-Start | Validate that the robot autonomously activates upon turning on the switch, testing motion, detection, and memory. |
| Line Accuracy | Ensure the robot consistently follows black tape mazes with no more than 5-line sensors. |
| Detection | Validate the sensor's detection state using an LED and monitoring its functionality. |
| Rotation | Observe the robot's ability to rotate, change direction, and maintain functionality on experimental mazes. |
| Account for Exits | Test if the robot can account for branching paths, returning to the main maze line as required. |
| Distance Recording | Verify the ultrasonic sensor's accuracy in measuring distances to objects from a detect line. |
| No Object Exception | Confirm the robot's behavior when encountering branched paths without objects or detect lines. |
| Memory | Monitor the robot's virtual memory to ensure it records distances, object placements, and the shortest distance. |
| Nearest Object | Check if the robot can return to the object with the smallest distance to its detect line on request. |
| Hardware and Software Integration | As a final test, simulate a "treasure hunt" in real-world conditions, evaluating overall system performance. |

## 2.5 Design Concepts and Justification

The functional requirements of the robot include the following:

1. Line Tracking

The robot must use sensors to detect the black line and be able to adjust its motion to follow the line.

The robot must also stop moving when it encounters a detect line.

1. Autonomous Navigation

The robot must perform all tasks without any intervention from the team once the treasure hunt has begun.

1. Maze Adaptability

The robot must be able to navigate any given maze layout.

The robot must be able to adapt to an unknown maze layout.

1. Object Detection

The robot must be able to detect objects placed in the maze.

1. Object Localisation

The robot must be able to measure the distance to each object from the corresponding detect line.

1. Memory

The robot must record the distance between each object and its detection line.

The robot must remember ‘where’ each object is placed within the maze.

The robot must be able to return to the object with the shortest distance to its detection line.

1. Start with Switch

The robot should begin its navigation of the maze when a switch is activated.

Focusing on the block diagram (2.1), submodules (2.2) interfacing (2.3) and test procedure (2.4) of the system, the best design choices were made.

The design can be simplified to each component having three connections. A signal connection to the Arduino, power, and ground connection. A ground plane was created using a separate Veroboard through which all components could easily be connected to ground. Logic level converters were placed centrally with the Arduino, as these components contained the most connections.

The PCB design was created with modularity in mind. Using a Veroboard, female connections for all component connections were soldered. Subsequent female connections aligning these pins were further soldered. Doing this allowed for complete modularity between all components on the Veroboard. Using male-to-male connectors, connections could be easily interchanged throughout debugging and testing. Whilst this method caused a convoluted design, the effectiveness of modularity outweighed the untidiness of the design.

The Veroboard provided an effective method for wiring the robot, as discussed above. However, a problem occurred when digital pin 8 (D8) was permanently set high. The underlying Veroboard solder had shifted to join two adjacent vertical connections. The choice was made to firmly break all connections between vertical lines on the Veroboard. Solely using male to male connections to interface components. Doing this meant all connections were easily visible, and no unwanted link was made between lines underneath the board.

For sensing and perception, the ultrasonic sensor needs to be positioned at an optimal height ang angle to detect objects accurately. The sensor, however, needs to be in an adequate position to allow for connection to the other submodules. The line sensors were calibrated and positioned to follow the designated path correctly. The two wide sensors (R2 and L2) were pushed to the maximum angle and distance away from the two central line sensors (R1 and L1) so branches and detect lines could accurately be detected.

The two Logic Level Converters were used as a buffer between the H-Bridge, Encoders and Ultrasonic Sensor connections to the Arduino. The Arduino pins connect to low level logic pins from which components are connected to the Arduino through the respective high level logic pins. Therefore, the signal voltage is stepped down to 3V3 from components to the Arduino during interfacing.

It was decided to not use the 5V output from the Arduino. The H-bridge contains a stable, voltage regulator circuit that outputs 5V from the battery. The Arduino was already powering the line sensors and logic level converters. To limit strain on the Arduino, all 5V lines were connected to the H-bridge.

Initially, two switches were used to turn on the robot. A main switch for providing power to the entire Veroboard, and a second switch, directly used for powering the Arduino. This first approach was effective because it allowed for the power and sensing subsystems to be turned off whilst code was being flashed on the Arduino. However, a significant power drop occurred when both switches were on. An initial 7V4 volts dropped to 5V5. Removing one of the switches meant the correct voltage from the batteries was transferred to the board when the robot was turned on. This meant other components were powered during the flashing of code, however, this process was quick and did not cause any damage to the system.

While the encoders could have been employed to steer the robot and assist in navigation, this approach was more complex. The Simulink blocks did not function as expected and caused error and overshoot. Using the outermost line sensors to trigger a state flow in MATLAB was more effective and easier to implement. The encoders were used minimally. Their main function was initially to measure and track distance, counting the number of ticks for every wheel rotation. This property became obsolete in our algorithms. The robot would never need to track its own physical distance. The number of branches were tracked and stored in the Arduino, allowing for the robot to return to paths with objects through memory, not the distance the robot had travelled to that branch.

The final design decision that was made towards the end of testing was the changing of Arduino pins. Initially, no analogue pins were used. The digital pins were sufficient in connecting components to the microcontroller and were easier to interface with from MATLAB. However, after exhaustive testing with these pins, some (for example D4) were damaged. The decision was made to connect line sensors to analogue pins in the event of a lack of digital connections. The line sensors did not connect through the logic level shifters, which allowed for simpler wiring to analogue pins. In MATLAB, the threshold for the analogue pins could further easily be set, causing the pins to function similarly to the digital connections.

# 3 Electrical Design

## 3.1 Power Calculations

Table 2: PCB Connections for Power

|  |  |
| --- | --- |
| **Component** | **Connection Pin** |
| Arduino | 7V4 – Battery |
| Ultrasonic Sensor | 5V – H-Bridge |
| Line Sensor 1 (Far Left Sensor) | 3V3 -- Arduino |
| Line Sensor 2 (Inner Left Sensor) | 3V3 -- Arduino |
| Line Sensor 3 (Inner Right Sensor) | 3V3 -- Arduino |
| Line Sensor 4 (Far Right Sensor) | 3V3 -- Arduino |
| Encoder 1 | 5V – H-Bridge |
| Encoder 2 | 5V – H-Bridge |
| Logic Level Converter (Low Voltage Pins) | 3V3 – Arduino |
| Logic Level Converter (High Voltage Pins) | 5V – H-Bridge |
| H-Bridge (Motor Driver) | 7V4 -- Battery |

Using the datasheets for each component, and factoring in connection types, the power for each component can be calculated using P = VI.

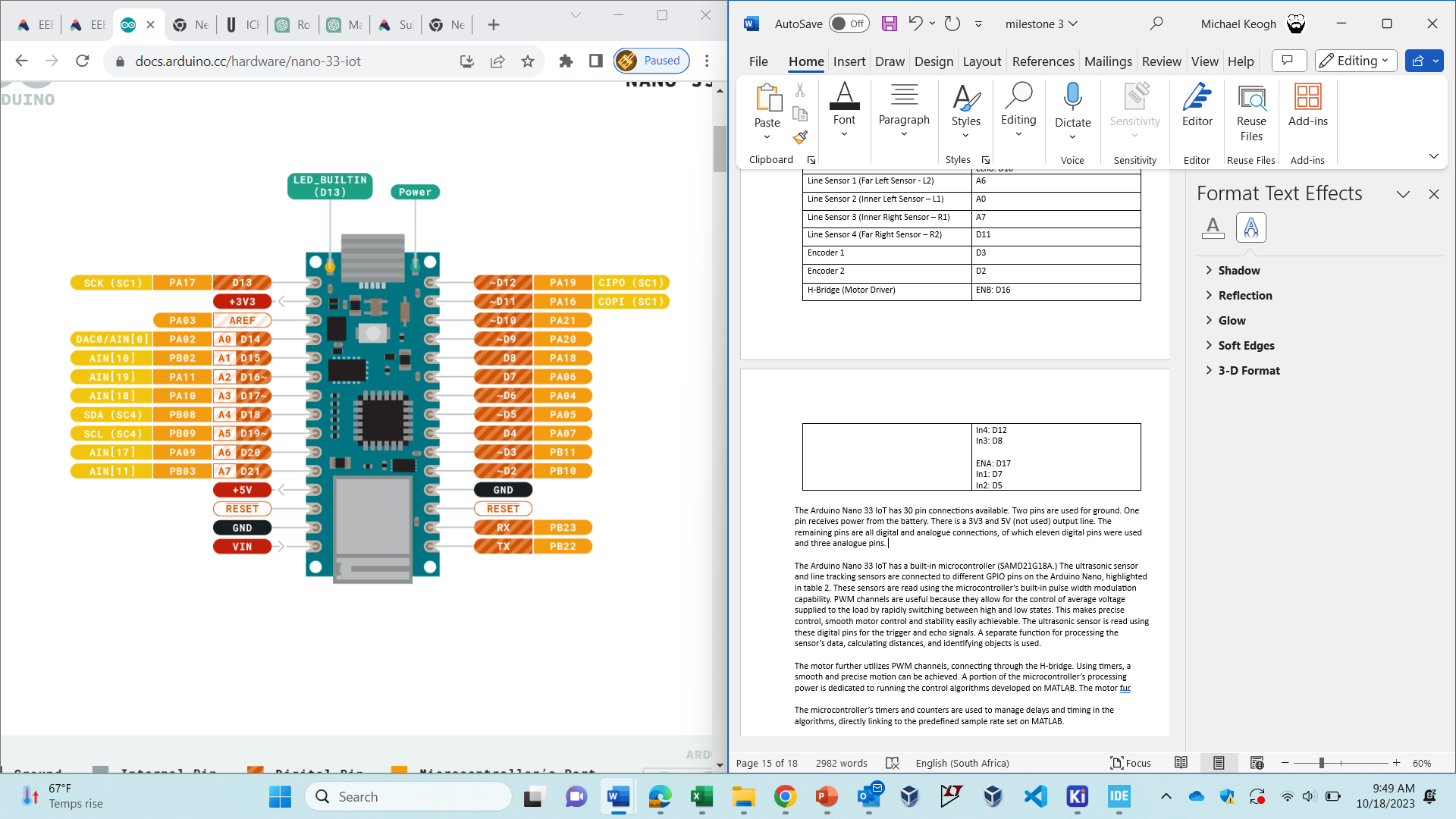
Table 3: Power Calculations

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Component** | **Supply** **Voltage** **(nominal** **V)** | **Minimum** **Current** **(mA)** | **Typical** **Current** **(mA)** | **Maximum** **Current** **(mA)** | **Minimum** **Power** **(mW)** | **Typical** **Power** **(mW)** | **Maximum** **Power** **(mW)** |
| Arduino | 5-18 (7.4 for robot) | 7 (per pin) | NA | 16 (per pin) | 51.8 | NA | 120 |
| Battery  (x2 18650 3V7) | 3.7 | NA | NA | 500 | NA | NA | 2850 |
| Ultrasonic Sensor | 5 | 15 | NA | 20 | 75 | NA | 100 |
| Line Sensor (x4) | 3.3 | NA | NA | 10 | NA | NA | 33 |
| Encoder (x2) | 5 | NA | NA | 20 | NA | NA | 100 |
| Logic Level Converter (High Level) | 5 | 0.1 | NA | 2 | 0.5 | 330 | 10 |
| Logic Level Converter (Low Level) | 3.3 | 0.1 | NA | 2 | 0.33 | 1 | 6.6 |
| H-Bridge | 7.4 | 1 | 20 | 120 | 4 | 40 | 240 |
| **Totals** | 19.4 | 11.5 | 827.51 | 2271.50 | 162.7 | 3070 | 12540 |

## 3.2 Microcontroller Resource Allocation

Table 4: PCB Connections for Arduino

|  |  |
| --- | --- |
| **Component** | **Connection Pin** |
| Ultrasonic Sensor | Trig: D9  Echo: D10 |
| Line Sensor 1 (Far Left Sensor - L2) | A6 |
| Line Sensor 2 (Inner Left Sensor – L1) | A0 |
| Line Sensor 3 (Inner Right Sensor – R1) | A7 |
| Line Sensor 4 (Far Right Sensor – R2) | D11 |
| Encoder 1 | D3 |
| Encoder 2 | D2 |
| H-Bridge (Motor Driver) | ENB: D16  In4: D12  In3: D8  ENA: D17  In1: D7  In2: D5 |

The Arduino Nano 33 IoT has 30 pin connections available, as depicted in figure 2. Two pins are used for ground. One pin receives power from the battery. There is a 3V3 and 5V (not used) output line. The remaining pins are all digital and analogue connections, of which eleven digital pins were used and three analogue pins.

*Figure 2: Arduino Nano IoT Pinout*

The Arduino Nano 33 IoT has a built-in microcontroller (SAMD21G18A.) The ultrasonic sensor and line tracking sensors are connected to different GPIO pins on the Arduino Nano, highlighted in table 2. These sensors are read using the microcontroller’s built-in pulse width modulation capability. PWM channels are useful because they allow for the control of average voltage supplied to the load by rapidly switching between high and low states. This makes precise control, smooth motor control and stability easily achievable. The ultrasonic sensor is read using these digital pins for the trigger and echo signals. A separate function for processing the sensor’s data, calculating distances, and identifying objects is used.

The motor further utilizes PWM channels, connecting through the H-bridge. Using timers, a smooth and precise motion can be achieved. A portion of the microcontroller’s processing power is dedicated to running the control algorithms developed on MATLAB. The motor fur

The microcontroller’s timers and counters are used to manage delays and timing in the algorithms, directly linking to the predefined sample rate set on MATLAB.

The built-in UART serial communication capabilities interface with external devices for debugging and data logging. Pins were set accordingly for this purpose. During stationary testing, the robot was turned on and connected to MATLAB. The state flow was observed, mapping which section of the code is being run for different scenarios. For example, looking at the state flow for line following, MATLAB was configured in build mode. Using black cardboard to represent the line, the sensing was tested, and conditions statements in the code were examined.

## 3.3 Power Design and Safety

Key:

7V4 Battery

3V3 Arduino

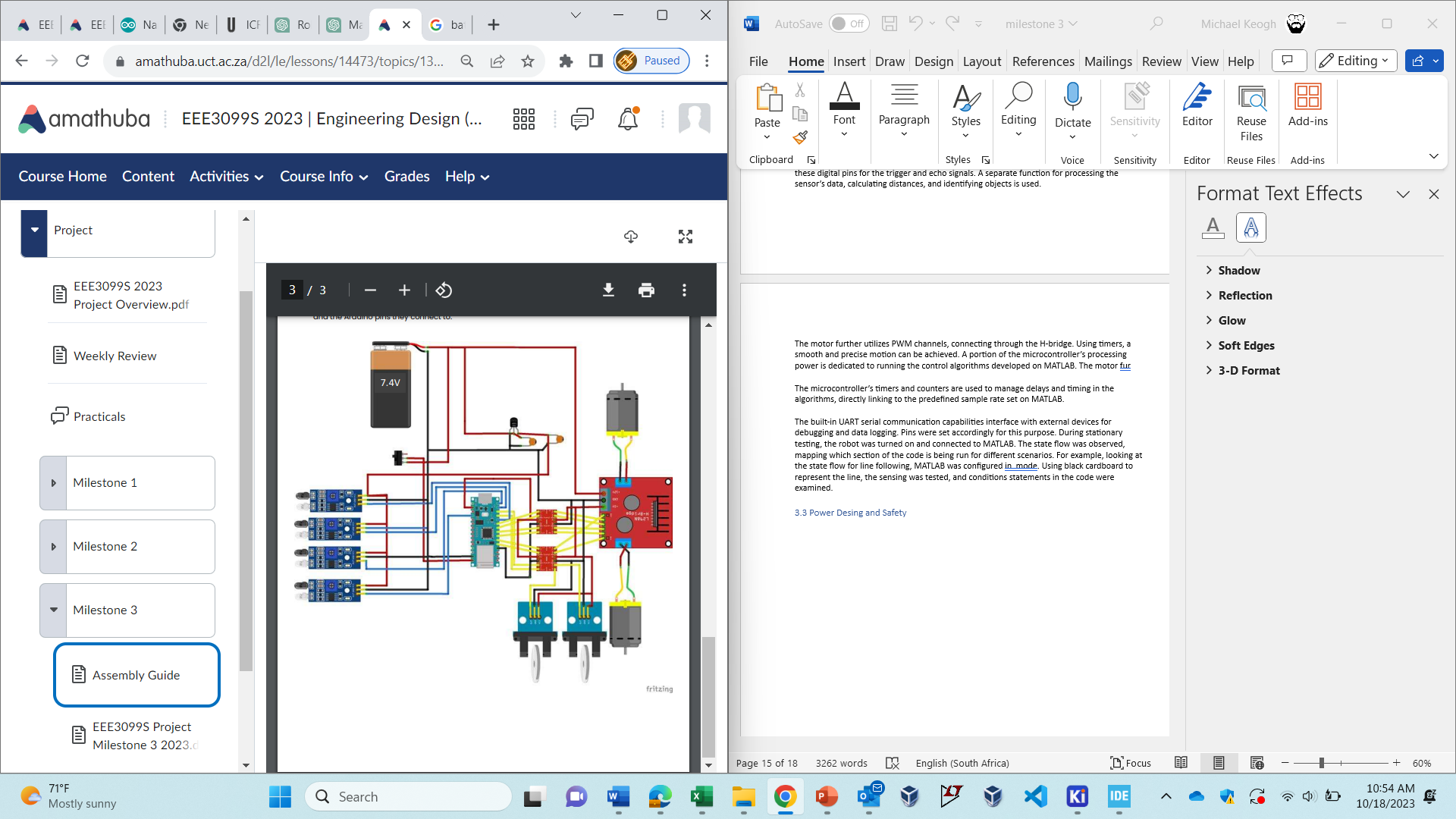
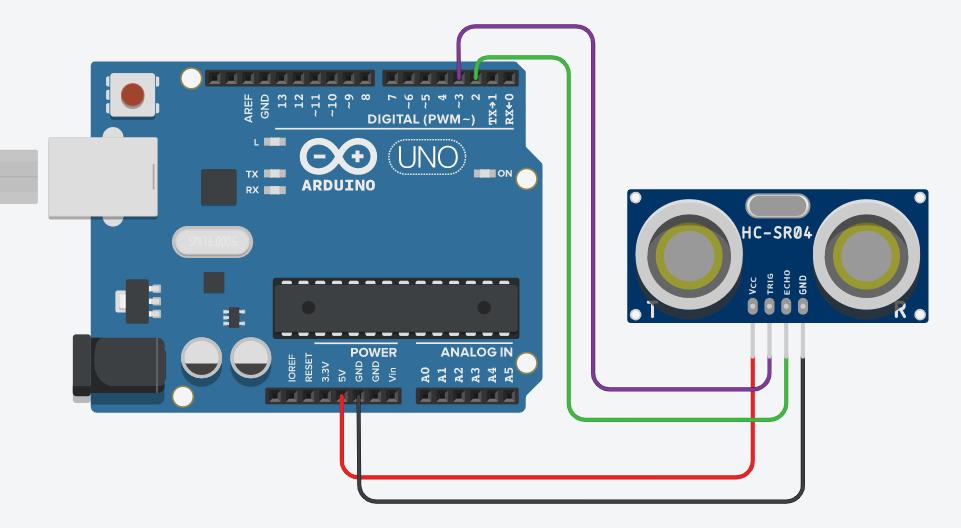
5V H-bridge

GND



### 3.3.1 Power Design

## 



*Figure 3: Power Design*

Figure 2 displays the power design on the Veroboard. The line sensors are linked to the 3V3 Arduino output line and the low-level logic side of the converters. The battery is connected through a switch, supplying 7V4 to the Arduino and H-bridge, which in turn powers the motors. The 5V output from the H-bridge connects to the high-level side of the logic converter, providing power to the ultrasonic sensor and encoders.

### 3.3.2 Power Safety

Implementing power safety mechanism in the autonomous mobile robot is crucial to prevent unexpected power-related issues. Before soldering the Veroboard, the circuit was built on a breadboard and voltage was tested at each component using a multi-meter.

On the soldered Veroboard, batteries were drained overtime. However, low battery voltage protection was easy to implement. The speed of the wheels for new 7V4 batteries was observed. When the wheel speed decreased significantly, this was indicative of a decrease in voltage and the batteries were changed. Additionally, at low voltage, the built in LED on the Arduino did not light up/was very dim during object detection which further indicated low voltage, meaning new batteries were needed.

Battery overcharge protection was initially absent in the design. This led to digital pins on the Arduino burning. To counteract any possibility of overcharge, the connections between all components on the vertical Veroboard lines were thoroughly broken. Tape was applied around any wire that had the potential of making an unwanted connection. Additionally, any solder was taped to prevent shorts, particularly at the switch pins. Male-to-male connectors are already isolated with insulation.

Importantly, the H-bridge and Arduino can accept a maximum voltage of 12V and 21V, respectively. The batteries will never exceed this value; thus a safe operating voltage is always guaranteed. The H-bridge does have a voltage regulator, outputting a steady 5V, and the Arduino outputs a constant 3V3 volts. Therefore, whilst a battery is not always a stable input supply, passing this voltage through the Arduino and H-bridge from which other components are powered, results in a stable voltage supply.

Emergency power cutoff was implemented using the switch provided. The switch can break the path of power to the entire Veroboard. Flipping the switch immediately disconnects power in the event of a malfunction.

Reverse polarity protection was included using a built-in diode. In the case of incorrect battery polarity connection, this component ensures current only flows in the correct direction, preventing the Arduino from being damaged.

Current limiting components were neglected because the individual IC’s have built in reactance’s.

Finally, as mentioned previously, using a set ground plane minimizes ground loops and noise.

## A screenshot of a computer program Description automatically generated3.4 Schematic

*Figure 4: PCB Design*

# 4 Motion Control

# 5 Line Sensing

# 6 Treasure Maze Solving

# 7 Conclusion

# 8 References

Core.ac.uk (2023) Available at: https://core.ac.uk/download/pdf/216953698.pdf (Accessed: 10 October 2023)

Docs.arduino.cc (2023) Available at: https://docs.arduino.cc/hardware/nano-33-iot (Accessed: 11 October 2023)

HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi (2023) Available at: https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi (Accessed: 14 October 2023)

Line\_Tracking\_Sensor\_for\_Arduino\_V4\_SKU\_SEN0017-DFRobot (2023) Available at: https://wiki.dfrobot.com/Line\_Tracking\_Sensor\_for\_Arduino\_V4\_SKU\_SEN0017 (Accessed: 11 October 2023)

Logic Level Converter Bi-Directional - Micro Robotics (2023) Available at: https://www.robotics.org.za/LEVEL-4P (Accessed: 14 October 2023)

Motor Driver Dual H-Bridge Module L298N - DIYElectronics (2023) Available at: https://www.diyelectronics.co.za/store/brushed-motor-drivers/34-motor-driver-dual-h-bridge-module-l298n.html (Accessed: 14 October 2023)

Owen-Hill, A. (2019) 'How to Solve Your Inspection Bottleneck With a Robot - RoboDK blog', RoboDK blog. Available at: https://robodk.com/blog/robot-inspection-bottleneck/ (Accessed: 14 October 2023)

Turtle: 2WD Arduino Mobile Robot Platform wiki- DFRobot (2023) Available at: https://wiki.dfrobot.com/2WD\_Mobile\_Platform\_for\_Arduino\_\_SKU\_ROB0005\_ (Accessed: 11 October 2023)

Wheel\_Encoders\_for\_DFRobot\_3PA\_and\_4WD\_Rovers\_\_SKU\_SEN0038\_-DFRobot (2023) Available at: https://wiki.dfrobot.com/Wheel\_Encoders\_for\_DFRobot\_3PA\_and\_4WD\_Rovers\_\_SKU\_SEN0038\_ (Accessed: 11 October 2023)