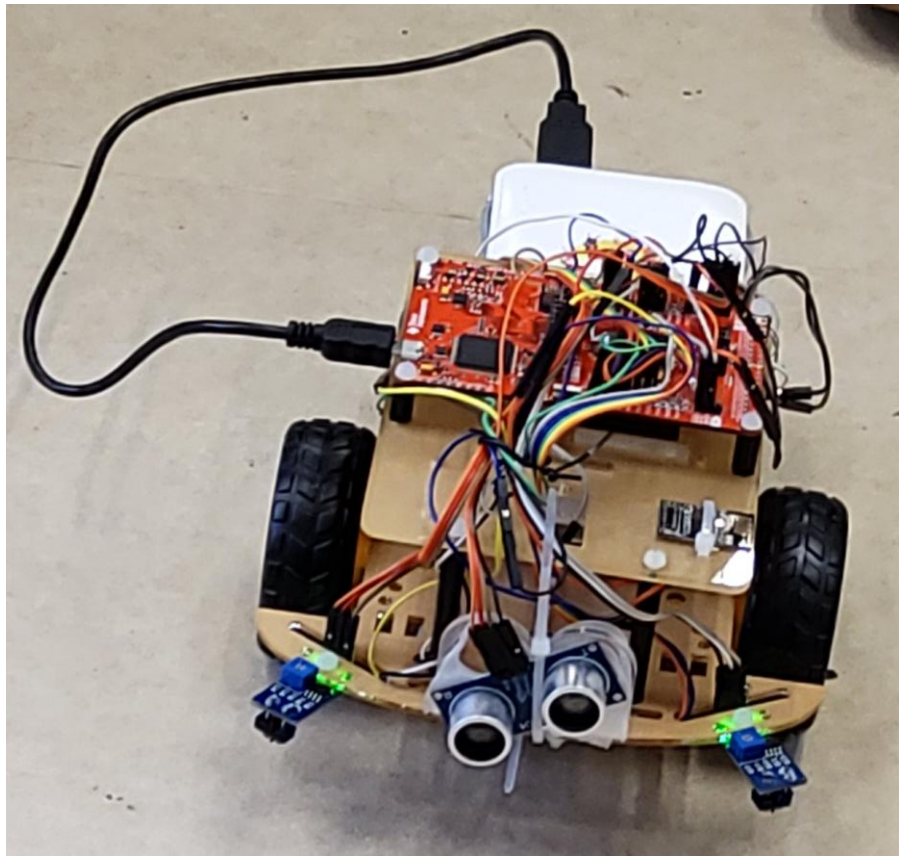


ROBOTIC CAR PROJECT BRIEF

Intelligent Autonomous Line-Following Robot

Background

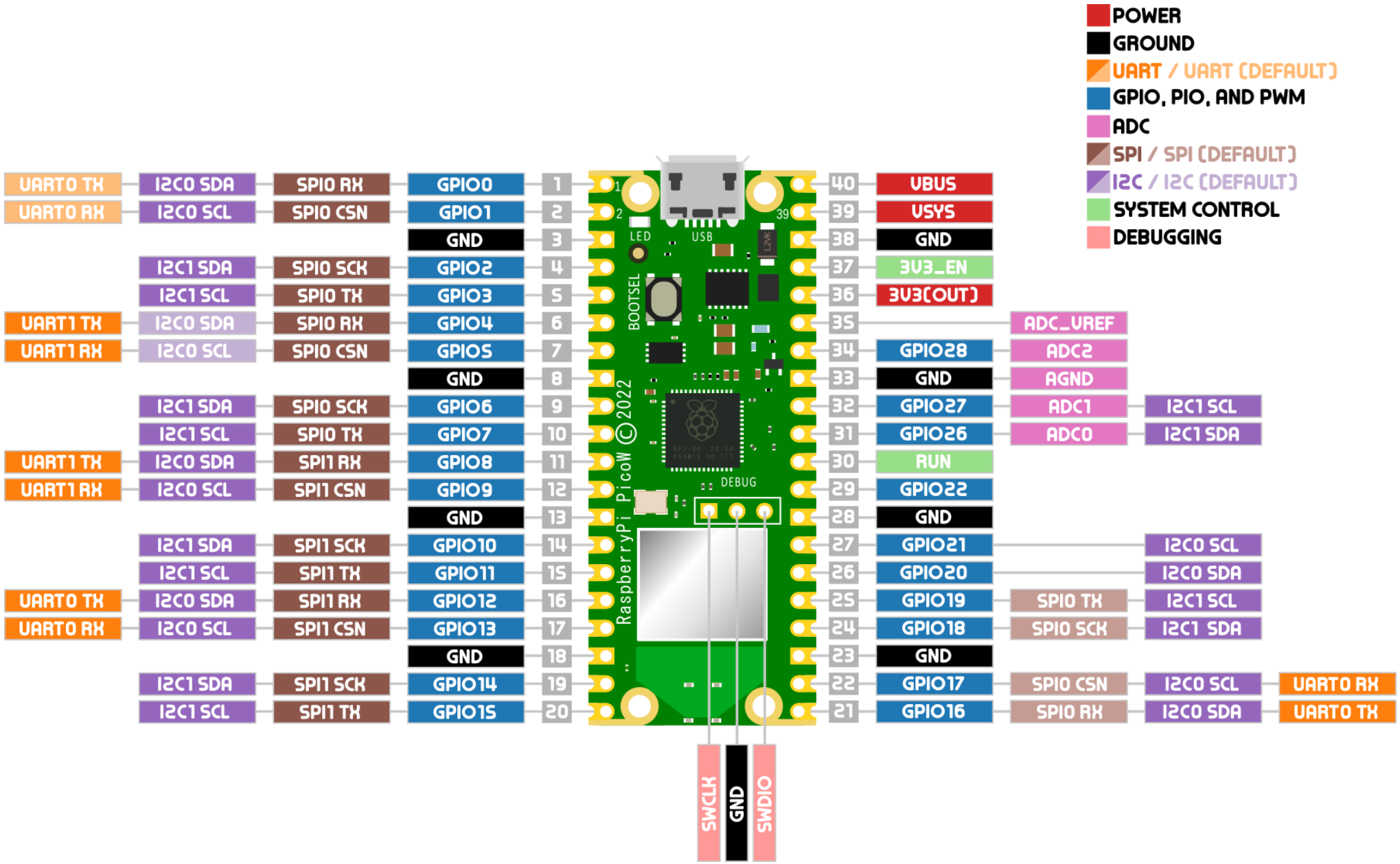
Line-following robots are a classic benchmark in embedded systems, testing real-time control, sensor fusion, and decision-making. This project extends the concept by integrating obstacle avoidance, encoded navigation commands, and remote telemetry, providing a realistic, multi-sensor embedded system challenge.

Objectives

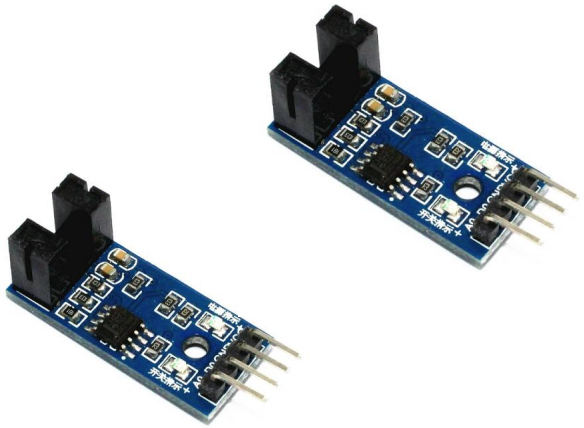
- Develop an autonomous robot capable of following a line, interpreting barcode commands, avoiding obstacles, and transmitting telemetry.
- Integrate multiple hardware components under real-time constraints.
- Optimise for embedded efficiency, ensuring accurate, reliable performance within limited resources.

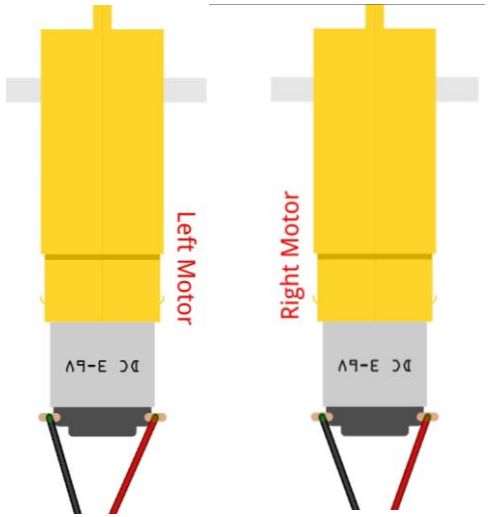
Key Features

1. **Line Following**: Single IR sensor with PID controller for precise tracking.
2. **Barcode Command Navigation**: Barcodes trigger specific route decisions (e.g., left/right turns) with IMU-assisted accuracy.
3. **Obstacle Detection & Avoidance**: Ultrasonic sensor and servo system detect and navigate around obstacles, temporarily leaving the line if required.
4. **Telemetry via Wi-Fi & MQTT**: Real-time data (e.g., speed, direction, distance) sent to a PC/mobile dashboard for monitoring and visualisation.

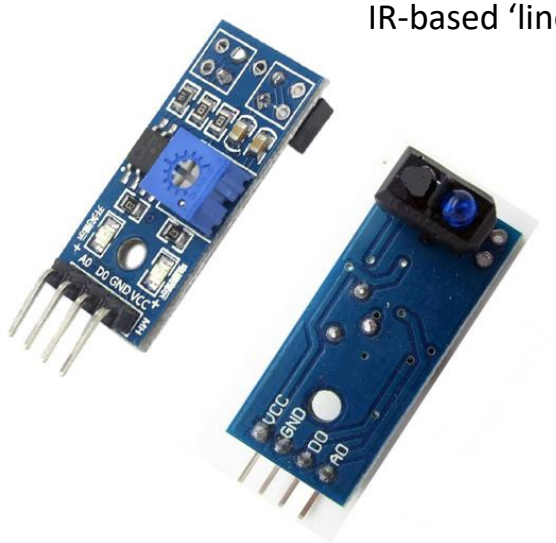

POWER
GROUND
UART / UART (DEFAULT)
GPIO, PIO, AND PWM
ADC
SPI / SPI (DEFAULT)
I2C / I2C (DEFAULT)
SYSTEM CONTROL
DEBUGGING
UART0 TX
UART0 RX
I2C0 SDA
I2C0 SCL
SPI0 RX
SPI0 CSN
GPIO0
GPIO1
1
2
GND
3
I2C1 SDA
I2C1 SCL
SPI0 SCK
SPI0 TX
GPIO2
GPIO3
4
5
UART1 TX
UART1 RX
I2C0 SDA
I2C0 SCL
SPI0 RX
SPI0 CSN
GPIO4
GPIO5
6
7
GND
8
I2C1 SDA
I2C1 SCL
SPI0 SCK
SPI0 TX
GPIO6
GPIO7
9
10
UART1 TX
UART1 RX
I2C0 SDA
I2C0 SCL
SPI1 RX
SPI1 CSN
GPIO8
GPIO9
11
12
GND
13
I2C1 SDA
I2C1 SCL
SPI1 SCK
SPI1 TX
GPIO10
GPIO11
14
15
UART0 TX
UART0 RX
I2C0 SDA
I2C0 SCL
SPI1 RX
SPI1 CSN
GPIO12
GPIO13
16
17
GND
18
I2C1 SDA
I2C1 SCL
SPI1 SCK
SPI1 TX
GPIO14
GPIO15
19
20
LED
USB
BOOTSEL
DEBUG
RaspberryPi PicoW ©2022
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
VBUS
VSYS
GND
3V3_EN
3V3(OUT)
ADC_VREF
GPIO28
GND
GPIO27
GPIO26
RUN
GPIO22
GND
GPIO21
GPIO20
GPIO19
GPIO18
GND
GPIO17
GPIO16
ADC2
AGND
ADC1
ADC0
SPI0 TX
SPI0 SCK
SPI0 CSN
SPI0 RX
I2C1 SCL
I2C1 SDA
I2C0 SCL
I2C0 SDA
I2C1 SCL
I2C1 SDA
I2C0 SCL
I2C0 SDA
UART0 RX
UART0 TX
SWCLK
GND
SWDIO

IR-based Wheel Encoder

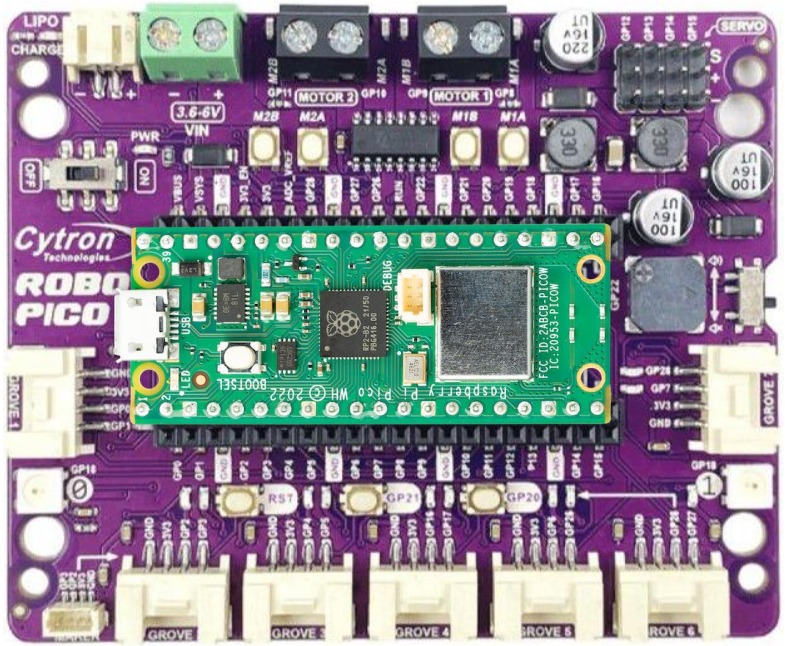




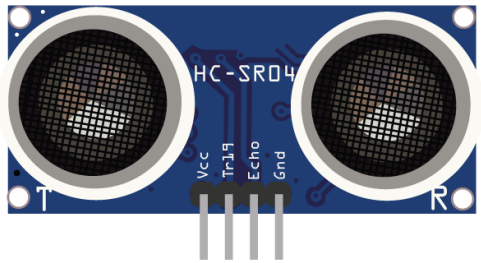
IR-based ‘line’ detector

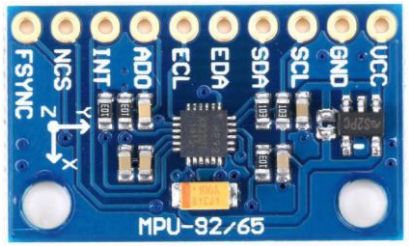


Motor Controller



Ultrasonic Sensor



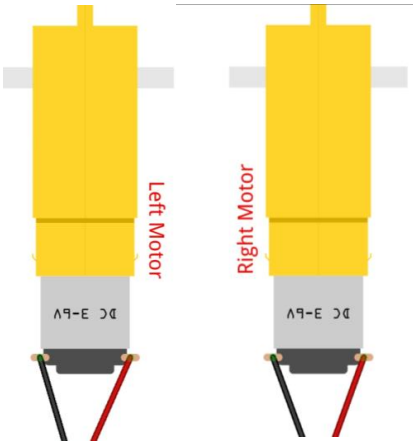


MP-9250
Accelerometer + Gyroscope + Magnetometer

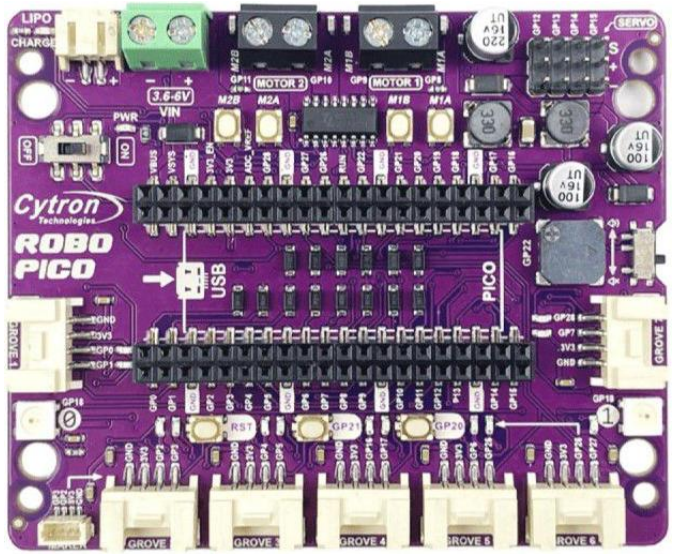
Servo

Motor Control System with PID Controller

- The motor control system drives the car's two wheels, allowing for **accurate** forward, backward, and turning movements.
- The PID (Proportional-Integral-Derivative) controller will fine-tune the motor's response, ensuring smooth and accurate movements.
- Use DC motors with the built-in motor drivers to control the wheels.
- IR-based wheel encoders attached to the wheels will provide feedback for the PID controller. (driver developed by another teammate)
- Implement a PID control algorithm that adjusts motor speed based on feedback from the encoders. This helps maintain a desired speed or position, correcting any deviations due to external factors like friction or load.
- PID tuning (adjusting the proportional, integral, and derivative gains) will be crucial to achieve the desired balance between responsiveness and stability.

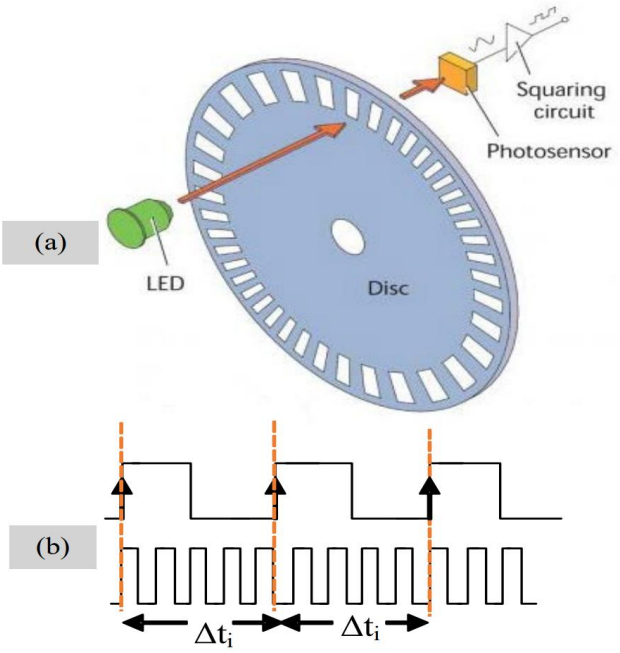


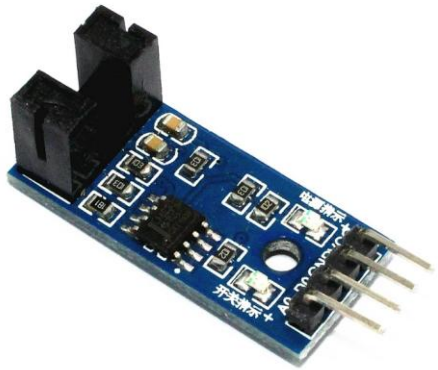
Motor Controller



Wheel Encoder for Speed and Distance Measurement

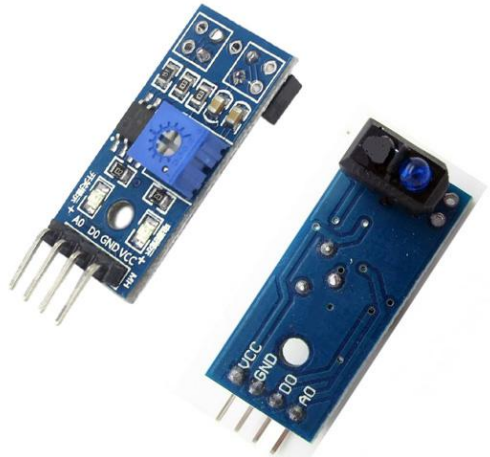
- Wheel encoders measure the rotation of the car's wheels, allowing for accurate tracking of **both speed and distance** travelled. This data is essential for precise control and navigation.
- Attach optical encoders to the wheels. These encoders generate pulses as the wheels rotate, which can be used to calculate the distance travelled.
    - The width (duration) of each pulse in the signal is directly related to the time it takes for a segment to pass the sensor. A longer pulse indicates slower rotation, and a shorter pulse indicates faster rotation.
- Develop algorithms to count the pulses from the encoders, calculate the car's speed, and determine the total distance travelled.
    - The PID controller will use this feedback to maintain the desired speed (done by motor team).
- Ensure the encoder signals are correctly interpreted and free from noise. Calibration may be needed to account for wheel slippage or other factors that could affect accuracy.





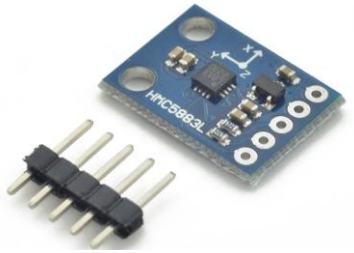
IR Sensors for Line Following and Barcode Decoding

- The IR sensors will enable the car to detect and follow lines on the ground, **typically** black lines on a white surface. Additionally, these sensors will decode barcodes placed along the line, allowing the car to interpret specific instructions or gather information.
- **Line Detection:** Mount multiple IR sensors beneath the car to accurately detect the line's position. These sensors will guide the car by adjusting its path based on real-time readings.
- **Calibration and Accuracy:** Ensure the sensors are properly calibrated to distinguish between the line and the background. The barcode decoder must be robust, capable of handling different barcode formats and incorporating error-checking mechanisms.
- **Barcode Decoding:** Use a dedicated IR sensor on the car's underside to scan barcodes located on the floor. The system will employ algorithms to analyse the sequence of lines detected by the sensor, interpreting the encoded information.
    - **Format:** Use the "Code 39" barcode standard. More information can be found here: https://www.keyence.com/ss/products/auto_id/codereader/basic/code39.jsp
    - **Robustness:** Evaluate the barcode reader's performance under various conditions:
        - **Bidirectional Scanning:** Can it decode the barcode when scanned in both directions?
        - **Variable Barcode Sizes:** Does it maintain accuracy with different barcode sizes?
        - **Speed Variations:** Can it reliably decode barcodes at different scanning speeds?



IMU Sensor for On-board Precision Control

- The robot will include an IMU sensor with a 3-axis accelerometer and a magnetometer. The IMU will now be mounted directly on the robotic car to enhance its movement accuracy and turning precision. The IMU will capture data on the robot's orientation, acceleration, and, optionally, magnetic field orientation. This data will feed into the car's control algorithms to achieve:
  - **Accurate Turns**: Using orientation data to align the robot correctly when following line segments or executing barcode commands.
  - **Stable Movement**: Monitoring acceleration to smooth out jerky motions and maintain consistent speed.
  - **Improved Navigation Decisions**: If the magnetometer is used, heading information can be combined with acceleration data for better directional control and stability.
  - **Data Filtering and Calibration to Noise Reduction**: Implement filters to remove noise from the accelerometer and optional magnetometer data, ensuring that only accurate, reliable data is used for control.

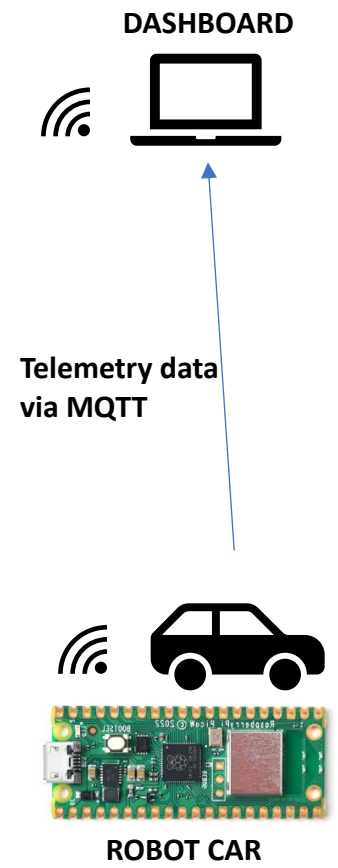


MQTT-Based Telemetry over WiFi

The robotic car will use a WiFi-enabled microcontroller to establish wireless connectivity with a PC or mobile dashboard. Instead of generic data transfer, the system will employ the MQTT protocol over WiFi for efficient, lightweight, and real-time telemetry communication. The telemetry data—including speed, distance travelled, sensor readings, and navigation status—will be published to an MQTT broker, where the PC or mobile dashboard can subscribe to relevant topics for live updates.

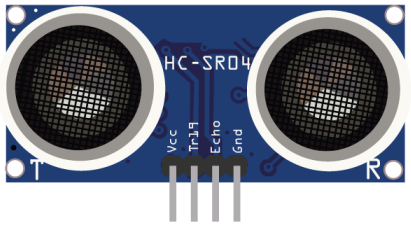
Key tasks include:

- **MQTT Integration:** Implement MQTT clients on the robotic car for publishing telemetry data and on the PC/mobile for subscribing to it. Robot Car should also be able to receive the barcode command for LEFT/RIGHT/STOP/U-TURN
- **WiFi Firmware Development:** Configure the microcontroller to maintain a reliable WiFi connection and manage MQTT messaging efficiently.
- **Data Structuring & QoS:** Use structured topic hierarchies and quality-of-service levels to ensure telemetry data remains accurate and timely.
- **Error Handling & Robustness:** Include reconnection logic and message validation to handle connection drops or data corruption.
- This approach ensures **low-latency, reliable, and easily scalable communication**, suitable for real-time monitoring and potential multi-robot setups in the future.



Obstacle Detection and Navigation with Ultrasonic + Servo

- The robotic car will use a **combination of ultrasonic sensing and a servo-mounted scanning mechanism** to detect obstacles blocking its path, measure the obstacle width, and plan a route around them.
- **Sensor Setup:**
  An ultrasonic sensor will be mounted on a servo motor at the front of the robot. Instead of pointing straight ahead only, the servo will rotate the sensor to **scan across a range of angles**, allowing the robot to map the obstacle's shape and width.
- **Width Measurement:**
  When an obstacle is detected, the servo will sweep the ultrasonic sensor left and right to identify the **edges of the obstacle**. The control system will use this data to estimate its width and determine whether there is enough space to navigate around it safely.
- **Path Planning:**
  Based on the obstacle map, the robot will:
  - Stop when an obstacle is detected within a critical safety distance.
  - Identify the clearer side with enough space for passage.
  - Temporarily leave the line-following path to move around the obstacle.
  - Rejoin the line after the obstacle is cleared.



- This approach ensures the robot not only **avoids collisions** but also **intelligently navigates around obstacles**, enabling fully autonomous operation on a cluttered track.

Report Assessment

- Use the template shared on xSITe
- All report submissions to the appropriate dropbox.
  - Week 5 – Requirements & Design
  - Week 9 – Detailed Design + Test Plan + Traceability
    + (include Test Results for five sub-systems)
    - Motor control with PID
    - Line following and barcode decoding
    - Detect distance from the obstacle and measure the width of the obstacle
    - Send data via MQTT
    - Obtain all data from IMU with data filtering
  - Week 13 – All Above + Test Result

Demo Assessment

- A demo schedule will be shared closer to the demo week
- All report submissions to the appropriate dropbox.
  - Week 6 – Individual Driver Demo
  - Week 10 – Partial Integration
  - Week 13 – Final Demo Run

Week 6 – Individual Driver Demo

1. Motor Controller + Wheel Encoder
   - No need for PID yet
2. IR Line sensor
   - can differentiate between black and white
   - measure the thickness of the black or white line
3. Ultrasonic + Servo
4. WIFI
   - No need for MQTT yet
5. IMU
   - No need for data filtering yet

Week 10 – Partial Integration

Demo 1: Basic Motion & Sensing Integration

Purpose: Validate motor control with PID, IMU data acquisition & filtering, and basic telemetry.

Setup:

- Place the robot on a simple straight track with **no** barcodes or obstacles or lines.
- Enable PID control for motor speed & heading.
- IMU sensor provides acceleration, orientation, and filtered heading to ensure straight motion.
- Telemetry (speed, distance, heading, raw vs filtered IMU data) published via MQTT to the dashboard.

Success Criteria:

- Robot maintains a straight path with minimal drift (thanks to IMU corrections).
- Live speed & heading data available on MQTT dashboard.
- Filtering visibly improves IMU data stability (less noise).

Demo 2: Line Following + Barcode Navigation + Telemetry

Purpose: Test full perception & decision loop with MQTT logging.

Setup:

- Closed loop track with **barcodes before junctions** to indicate commands (LEFT/RIGHT/STOP).
- Robot follows the line using IR sensor + PID.
- Barcodes trigger route decisions; IMU ensures accurate turning angles.
- Telemetry publishes: barcode command, current state, speed, distance travelled.

Success Criteria:

- Robot correctly follows the line, interprets barcodes, and executes correct movements.
- MQTT dashboard shows **real-time line position, barcode events, and movement states**.
- No track loss even after multiple junctions.

Demo 3: Obstacle Detection + Avoidance + Telemetry

Purpose: Validate ultrasonic + servo obstacle detection, width measurement, avoidance, and full telemetry stack.

Setup:

- Track with one obstacle blocking the path.
- Servo-mounted ultrasonic scans left/right to measure obstacle width & identify clear side.
- ~~Robot detours around the obstacle and rejoins the line; IMU ensures smooth turns.~~
- MQTT publishes obstacle width, ~~clearance path~~, and telemetry logs.

Success Criteria:

- Robot stops at obstacle, scans edges, ~~picks correct side, navigates around it, and rejoins line~~.
- MQTT dashboard shows obstacle data, ~~chosen path, and recovery status in real time~~.
- Final log captures speed, distance, IMU data, line events, and obstacle encounters.

Week 13 – Final Demo Run

- Max 3-tries outcome-based
- Latest grade taken and not the highest. (Thus, test, test and test before you get assessed)