CAV Model Development Platform Plan for Cyber Security Testing and Resilience

Justin Atkins 17831226

CAV Project Development:

Staging Plan

(Complete) Phase 1: Construction and Components.

(Complete) Requirement Analysis: Define the specific requirements for your CAV's software, including its behaviour, safety features, communication protocols, and user interactions. Consider both functional and non-functional requirements.

(Complete) Point of hardware Control and Powertrain Dynamics: Design and configure power train to respond to external commands from MCU control Hardware and ensure physical modelling and functionality.

(Complete) Assembly and Configuration of CAV Componentry: Sensory Hardware, powertrain and power supply hardware has been fitted to CAV module and functional parameters are ready for determination. The module has been preconfigured using Ubuntu 20.04 through the Linux 5.10 kernel. It is an AARCH64 system processor and has an inbuild CPU MCU and GPU all configured with the necessary drivers for associated hardware.

(Complete) Sensor Integration: Integrate the ultrasonic, radar, LiDAR, and stereo camera sensors with your CAV's control system. Develop the necessary interfaces to collect data from these sensors.

(Complete) Data Collection and Annotation: Collect real-world data from the sensors installed on the physical car. This data will be used for training and validating your autonomous driving algorithms. Annotate the data to label objects, lanes, and other relevant information.

Phase 2:

Simulator Development: Create a simulated environment that replicates the real-world conditions your CAV will encounter. This simulated environment will be used for training and testing your autonomous driving algorithms.

Sensor Simulation: Integrate sensor simulation models into the simulator to mimic the behaviour of ultrasonic, radar, LiDAR, and stereo camera sensors. This will provide synthetic sensor data for training and validation.

Algorithm Development: Develop the autonomous driving algorithms using machine learning techniques, such as deep learning and computer vision. These algorithms should be capable of processing sensor data to make decisions about steering, braking, and acceleration.

Model Training: Train your machine learning models using the annotated real-world and simulated data. Use techniques like supervised learning and reinforcement learning to improve the algorithms' performance.

Phase 3:

Testing and Validation: Test the algorithms extensively in both the real world and simulated environments. Verify their accuracy, robustness, and safety. Iteratively refine the algorithms based on the testing results.

Safety Measures: Implement safety measures such as collision avoidance, emergency braking, and fail-safe mechanisms to ensure the CAV can respond appropriately to unexpected situations.

User Interface Development: Create a user interface that allows users to interact with the CAV's autonomous features, monitor its status, and take control when necessary.

Deployment: Deploy the finalized software onto the physical CAV. Conduct thorough testing in controlled environments before releasing the CAV for real-world testing.

Final Phase:

Develop Interconnection Medium for Multiple CAVs: Provide a real time server where multiple CAVs will be able to communicate with one another exchanging on road details about their surrounding environment and positional data for exchange between modules.

Real-World interconnection Testing: Conduct extensive real-world testing of the autonomous capabilities of the CAV in various scenarios, including urban, suburban, and highway driving.

Continuous Improvement: Gather data from real-world testing and user feedback to continuously improve the software. Update the algorithms to enhance safety, efficiency, and user experience.

Maintenance and Updates: Regularly update the software to address bugs, security vulnerabilities, and to incorporate advancements in autonomous driving technology.

NOTE:

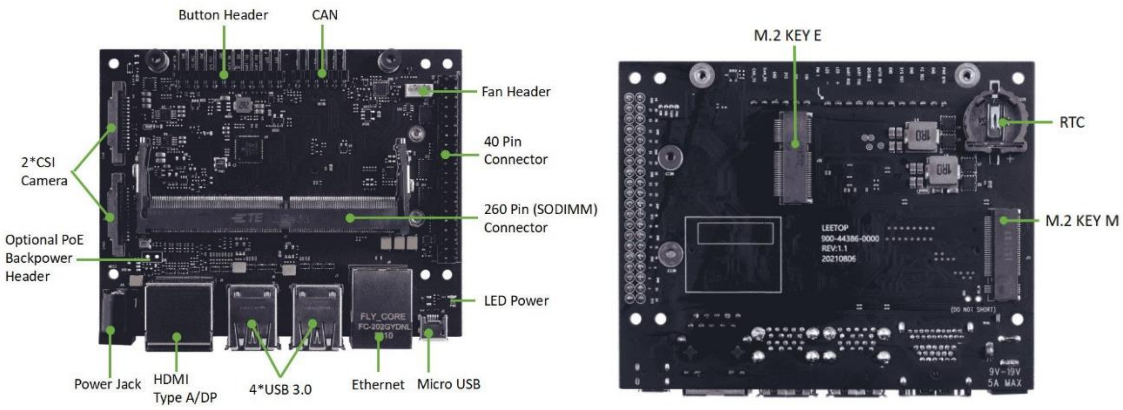
It Would be helpful for conceptual understanding if you guys were to read up on the modules, we will be utilising within our code, and the functionality of the hardware involved. This is purely an exercise to informal all members of the underlying control and sensory hardware being utilised and their proficiencies and shortfalls. **If you have any questions regarding any of the below information, please do not hesitate to ask me. As the understanding of how the Software will translate to the physical model is vital.**

Background Information and Initial configuration and design:

The testing configuration includes integrated circuits that employ multiple methods for capturing sensory data. Research into the industry standard sensors for Connected and Autonomous Vehicles (CAV) has revealed that the primary sources of sensory input revolve around LiDAR, computer vision using stereo cameras, ultrasonic and Radar perception. Integrating these systems empowers CAVs to comprehend their surroundings by utilizing these sensory devices, resulting in the creation of a three-dimensional map of the vehicle's environment. The presence of multi-channel sensing is critical for CAVs, as their ability to navigate their surroundings hinges on the accuracy and availability of information. Multi-channel sensing enables CAVs to validate and adjust their perception of their state based on information from multiple sources rather than relying on a single source. This enhances the CAV's robust spatial awareness and allows a range of sources to be employed to validate its condition relative to environmental circumstances.

Computational Hardware:

Seeed Studios J2022:



The reComputer J2022 is a hand-size edge AI box built with Jetson Xavier NX 16GB module which delivers up to 21 TOPs AI performance, a rich set of IOs including USB 3.1 ports(4x), M.2 key E for WIFI, M.2 Key M for SSD, RTC, CAN, Raspberry Pi GPIO 40-pin, and so on, aluminium case, cooling fan, pre-installed JetPack System, as NVIDIA Jetson Xavier NX Dev Kit alternative.

Features:

Hand-size edge AI device: built with Jetson Xavier NX 16GB Production Module, delivering 384 NVIDIA CUDA® cores delivers up to 21 TOPs AI performance with a small form factor of 130mm x120mm x 50mm.

NVIDIA Jetson Xavier NX Dev Kit alternative: the carrier board brings rich IOs including a Gigabit Ethernet port, 4 USB 3.1 ports, an HDMI port, and a DP port.

Pre-installed NVIDIA JetPack: support the entire Jetson software stack and various developer tools for building fast and robust AI application provided by Seeed Edge AI partners, helps develop innovative AI solution for manufacturing, logistics, retail, service, agriculture, smart city, healthcare, and life sciences.

Specifications:	
Brand:	reComputer
Model:	J2022-Edge
Power Requirements:	19V / 4.75A DC
UBS:	4x 3.1 Type A Ports, 1x USB-C
M.2 Key Supports:	WIFI, BLE, SSD, RTC, CAN, Raspberry Pi GPIO 40 pin
Module:	Jetson Xavier NX 16 GB
GPU:	384-Core NVIDIA Volta
CPU:	6-Core NVIDIA Carmel ARM, V8.2 64-bit CPU 6 MB L2+ 4 MB L3
Memory:	8GB 128bit LPDDR4x
Storage:	16GB eMMc 5.1 + SDD Expansion
Video Encoder:	2x4K60, 4x4K30, 10x1080p60
Video Decoder:	6x4K60 22x1080p60
Display:	HDMI Type A
Camera:	2*CSI Camera (15pos, 1mm pitch, MIPI CSI-2)
Dimensions:	130x120x50mm

JetPack SDK

NVIDIA JetPack SDK is the most comprehensive solution for building end-to-end accelerated AI applications. JetPack provides a full development environment for hardware-accelerated AI-at-the-edge development on Nvidia Jetson modules. JetPack includes Jetson Linux with bootloader, Linux kernel, Ubuntu desktop environment, and a complete set of libraries for acceleration of GPU computing, multimedia, graphics, and computer vision. It also includes samples, documentation, and developer tools for both host computer and developer kit, and supports higher level SDKs such as DeepStream for streaming video analytics.

The J2022 has gained an additional 500GB NVME memory expansion and is booting natively from the expanded memory: The Computer has the Latest Version of Jetpack 5.1.2 and is being run on the 5.10 Linux Kernel with ubuntu 20.04.

Please see: https://developer.nvidia.com/embedded/jetpack For more specific details.

Overview of Sensors and Relative Perception:

Stereo Vision Camera:

Stereo cameras are a type of camera setup that mimics human binocular vision to perceive depth and three-dimensional (3D) information from the surrounding environment. They work by capturing two images of the same scene from slightly different viewpoints, similar to how our eyes perceive objects from slightly different angles. The sensory acquisition is achieved through a fixed focal length predetermined by the manufacturer in order to set the Focal distance (the distance between our eyes) to render a 3D object from two 2D images). The camera will primarily used of object detection and road feature recognition. This will be vital for our pathway planning as the stereo camera will be the only sensor capable of detecting on road signage and lanes.

Relevant information and Source documentation to get familiar with.

- **Object Detection (detectnet.py)**
  - detectNet - The NN object accepts an image as an input and outputs a list of coordinates of the detected bounding boxes along with their classes and confidence values. The platofrm is able to be used under C++ and Python where various pretrained models are available.
  - The chosen pretrained network should be tailored for CAV related classification. Models such as TAO DashCamNet and TOA TrafficCamNet should be favoured due to their relevance to CAV applications. The classification objects within the models both contain 4 classes (person, car , bike and sign).

- **Semantic Segmentation**
  - SegNet.py - based on image recognition, the classification of objects within images occurs at the pixel level and based on defining a region within an image. The boundaries are defined through concolutionalizing a pre-trained image recognition model which utilises a Fully Convolutional Network (FNC) capable of per-pixel labelling.
  - The architecture yields useful environmental perception through pixel based classification of many various objects including fore & background.
  - The below example uses the pre-trained Cityscapes Classification model. The model is specialised to discern objects found under road going conditions in urban environments. More test images for training can be found within the training data set: **city-*.jpg**
  - Promising test results were found utilising the jetson Xavier on the resnet18 architecture.

Open-Source Autonomous Driving Models:
https://github.com/udacity/self-driving-car/blob/master/README.md
Road Sign Detection:
https://www.kaggle.com/datasets/andrewmvd/road-sign-detection

Navigation paths and Pathway Planning:
https://universe.roboflow.com/browse/self-driving

Stereo Camera used: ORBBEC Astra 3D

Designed to capture both colour and depth information simultaneously.

Key Features:

Depth Sensing: The Orbbec Astra camera uses active depth sensing technology, typically based on structured light or time-of-flight principles, to capture accurate depth information of the surrounding environment. This allows it to create detailed 3D point clouds or depth maps.

RGB Imaging: In addition to depth sensing, the camera also captures RGB (color) images of the scene. By combining depth and color information, the Astra camera provides a comprehensive representation of the environment.

Structured Light or Time-of-Flight: Different models of the Orbbec Astra camera might use different depth sensing technologies. Structured light involves projecting a pattern of light onto the scene and measuring how the pattern is distorted by objects, while time-of-flight measures the time it takes for light to travel to objects and back to the sensor.

Wide Field of View: The Astra camera often features a relatively wide field of view, which is beneficial for applications requiring a broad perspective of the environment.

SDK and Software Support: Orbbec provides a software development kit (SDK) and software tools that enable developers to access the camera's data streams, process depth and color data, and build applications that leverage the 3D sensing capabilities.

Compatibility: The camera is often designed to work with common development platforms like Windows, Linux.

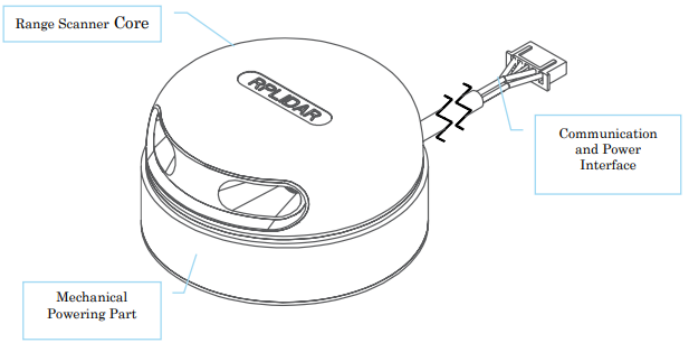
Slamtec A2M12 LiDAR:

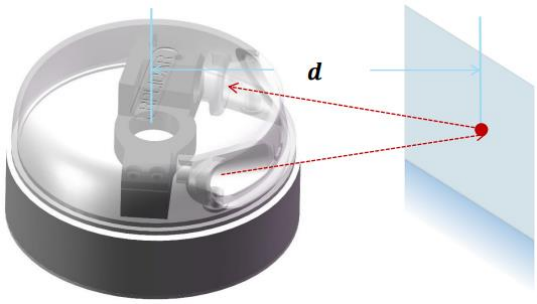
LiDAR systems preform 2D scan of its environment pulling in distance data from its surrounding environment. The A2M12 has a range accuracy of <2% between 0.2-10m and <2.5% between 10-16m. The typical scanning frequency of the module is 10Hz @600rpm with a sample rate of 16KHz. This places the modules angular resolution at 1sample for every 0.225° of rotation. The A2M12 is suitable for indoor, outdoor and low light operation and satisfies laser eye safety regulation IEC-60825 with beam properties of:

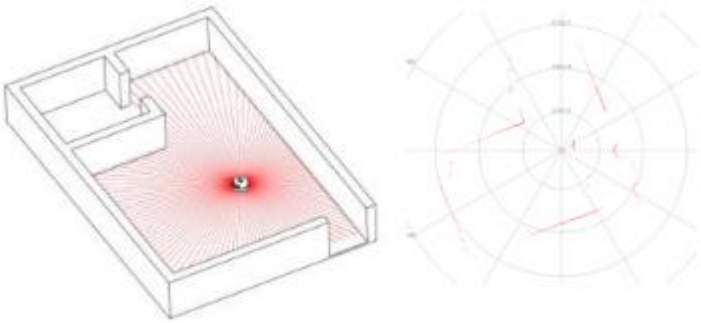
Item	Unit	Min	Typical	Max	Comment
Laser Wavelength	nm	775	785	795	IR Beam
Laser Power	mW		10	12	Peak Power
Pulse Length	us	60	87	90	Pulse Time
Laser Class	IEC	60825	60825	60825	Laser Class

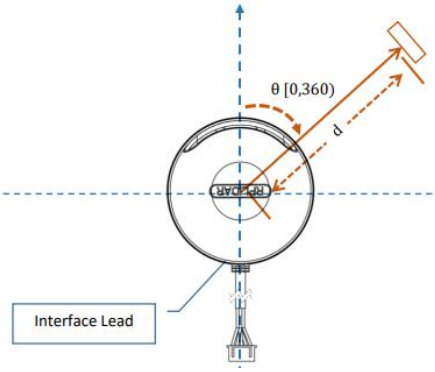
The beam emission is passed from the scanner core and is timed awaiting its arrive back at the scanner core where, compared against the speed of light c= $299.79 \times 10^6 m/s^2$ as to calculate the distance of surrounding objects from the scanner. The embedded RPLIDAR will

process the sample data and will output the distance and angle values between the object and the module via the communication interface.

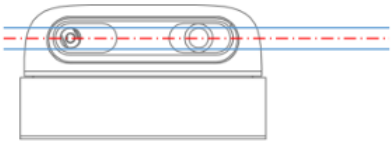


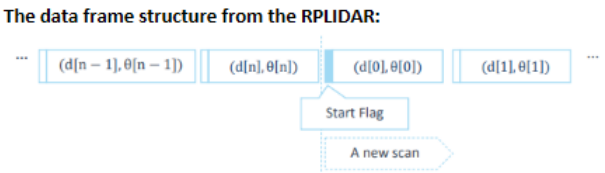




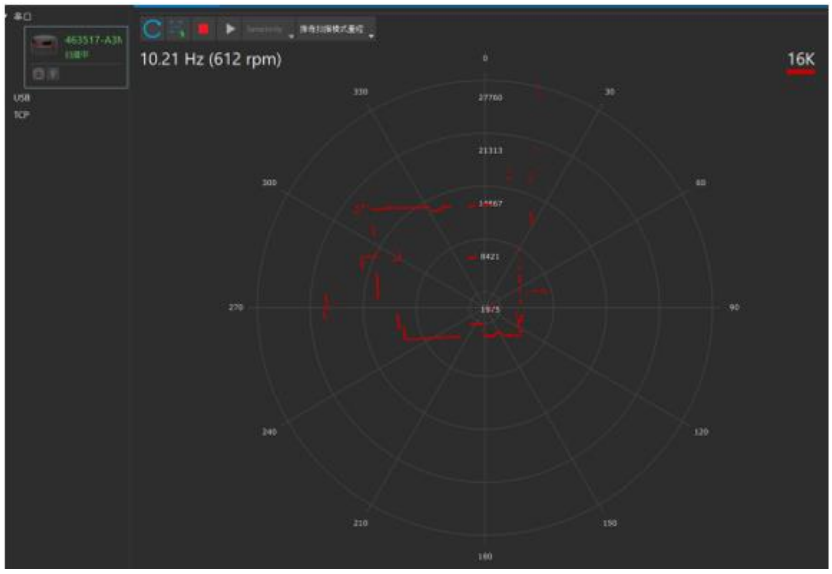


Data Type	Unit	Description
Distance	mm	Current measured distance value between the rotating core of the RPLIDAR and the sampling point
Heading	degree	Current heading angle of the measurement
Start Flag	(Bool)	Flag of a new scan
Checksum		The Checksum of RPLIDAR return data
Communication Speed	bps	256000
Communication Interface	Serial	TTL UART
Compatibility		Supports former SDK protocols





SDK and support: To utilise the RDLIDAR module a product development platform is a available to speed up development. The frame grabber plugin for Robo Studio is available for testing and debugging as well as the SDK available under Windows x86 and Arm Linux.



Global Positioning System (GPS): A satellite-based navigation system that allows for accurate determination of position, velocity, and time anywhere on or near the Earth's surface. In the context of Connected and Autonomous Vehicles (CAVs), GPS technology plays a role in aspects of navigation, control, and overall functionality.

Position and Localization: GPS provides accurate geographic coordinates (latitude, longitude, and altitude) to determine the precise position of the CAV on the Earth's surface. This information is essential for the vehicle to understand its location within the road network and navigate to a desired destination.

Waypoint Navigation: CAVs can use GPS waypoints as targets to guide their path. By setting waypoints, the vehicle can follow a predetermined route, avoid obstacles, and accurately stay on track.

Map Matching: GPS data can be used to match the vehicle's position to a digital map of the road network. This enables the CAV to understand which lane it's in, upcoming intersections, and road geometries. Map matching enhances the accuracy of the vehicle's position and helps in proper lane keeping.

Route Planning: GPS data combined with map information allows the CAV to plan optimal routes to reach its destination. The vehicle can consider real-time traffic conditions, road closures, and other factors to choose the fastest and most efficient route.

Collision Avoidance: GPS data can contribute to collision avoidance systems. By knowing the precise positions of nearby vehicles, the CAV can anticipate potential collisions and adjust its behaviour accordingly.

Localization Redundancy: While GPS is accurate, it might experience inaccuracies or disruptions in certain environments (e.g., urban canyons, tunnels). In such cases, CAVs often

use additional sensors like LiDAR, cameras, and radar to improve localization accuracy and redundancy.

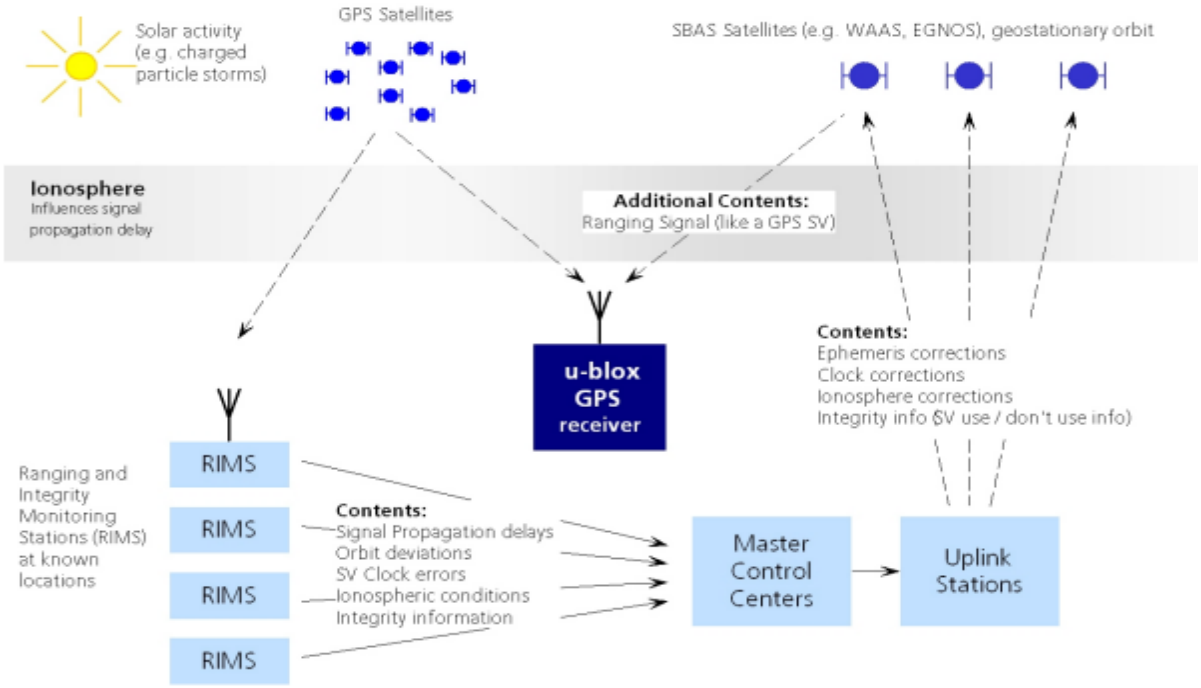
Highway Driving and Lane Changes: GPS is particularly useful for highway driving and lane changes. The CAV can use GPS information to navigate smoothly on multi-lane highways, change lanes safely, and handle complex highway interchanges.

Geofencing and Operational Boundaries: GPS can be used to define geofences, which are virtual boundaries on a map. CAVs can be programmed to respect these boundaries, enabling features like restricted zones, speed limit enforcement, or controlled areas (e.g., school zones).

Remote Control and Monitoring: Fleet managers and operators can use GPS to remotely monitor the status and location of CAVs. This is useful for tracking vehicle movements, managing routes, and ensuring compliance with operational plans.

Precise Stop Locations: When the CAV reaches its destination, GPS can help ensure precise stop locations, which are important for passenger pickup/drop-off, parking, and docking with charging stations (if applicable).

Updating Maps and Navigation Data: GPS-enabled CAVs can contribute to mapping and navigation data by sharing real-time traffic information and road conditions, helping to improve the accuracy of navigation systems for all vehicles.

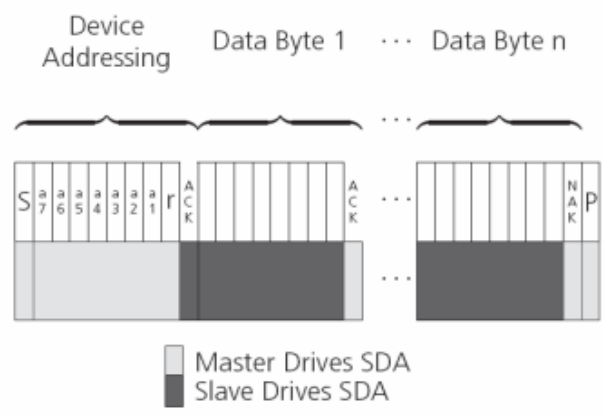


7.5.1.2 Current Address Read

The receiver contains an address counter that maintains the address of the last register accessed, internally incremented by one. Therefore, if the previous read access was to address `n` (where `n` is any legal address), the next current address read operation would access data from address `n+1` (see Figure *DDC Current Address Read Access*). Upon receipt of the device address with the `RW` bit set to one, the receiver issues an acknowledge and the master can read `1` to `N` bytes from the receiver, generating a not-acknowledge and a stop condition after the last byte being read.

To allow direct access to streaming data, the internal address counter is initialized to 0xFF, meaning that current address reads without a preceding random read access return the raw message stream. The address counter can be set to another address at any point using a random read access.

DDC Current Address Read Access

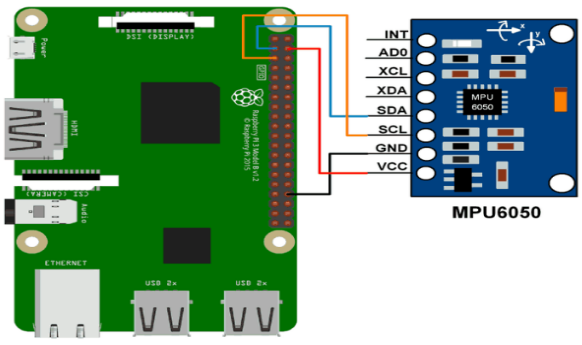


Inertial Measurement Unit (IMU): An electronic device that utilising microscopic sensors to determine present forces active on it in the form of acceleration, velocity, position and orientation. The inbuilt 6050 Unit present in the sensor features 6-axis tracking relative to the x,y,z-axis. The axis's described allow for the interpretation of linear and rotational data in linear-angular acceleration, velocity, and displacement.

The unit contains an inbuilt Digital Motion Processor (DMP) capable of processing complex 9-axis Motion Fusion algorithms which extend to inputs of outboard magnetometers and other applicable sensors via I2C bus.

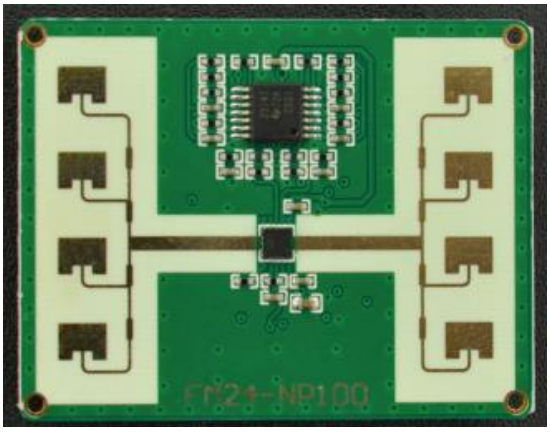
The 6 Dof sensor breakout integrates with the MPU6050 sensor and a low noise 3.3V regulator and pull up resistors for the I2C bus. So its available to directly hookup to the sensor via a MCU I2C comms.

Connection Diagram of MPU6050 with Raspberry Pi



MPU6050 Interfacing with Raspberry Pi

24GHz Microwave Radar

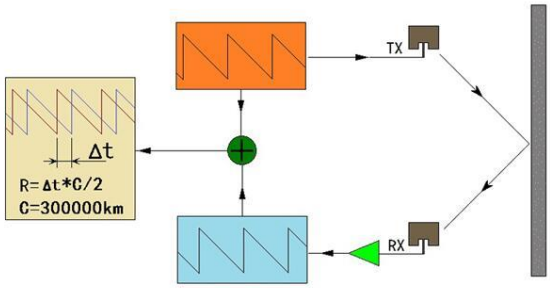


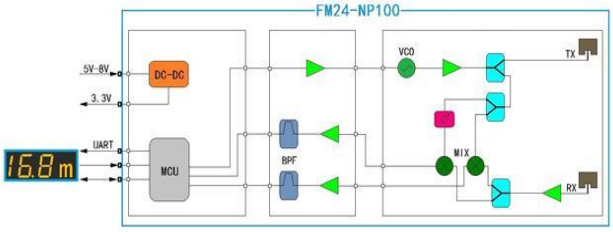
Microwave radar sensor can detect distance by transmitting and receiving radio waves. Compared with range & distance sensor of other kinds, the radar sensor can provide much smaller size, lighter weight and wider detection range. Besides, the sensor can keep a steady performance even operated in harsh environments and be able to penetrate most non-metallic materials, dust, smoke and fog etc.. What's more, it can be used to detect distance of multiple targets.

The function of a mm distance sensing radar is to allow robust forward face object detection at a max distance of 30m.

Features

- Non-contact detection
- Impervious to temperature, humidity, noise, airflow, dust, light and so on, suitable for harsh environment
- Strong anti-interference ability
- Low power output, saft and environmentally friendly
- Long detecting distance, high accuracy
- Large sensing angle, wide detecting range
- Support distance detection of multiple targets
- High directivity, travel at the speed of lightc





Specification

- Input Voltage: 4-8V（DC）
- Input Current: >100mA
- Dimension: 34mm×44mm×5mm/1.34×1.73×0.20”
- Weight: 8g
- Storage Temperature: -40℃~80℃
- Operating Temperature: 0℃~70℃
- **Transmit:**
  - Detection Distance：0.5~20m
  - Frequency: 24GHz
  - Power: 6dBM(Max 10dBM)
  - Modulation: FMCW
- **Receive:**
  - Data Update Speed: 10Hz
  - H—Plane Half Power Beamwidth: 78（-3db）
  - V—Plane Half Power beamwidth: 23（-3db）

Ultrasonic Sensors:

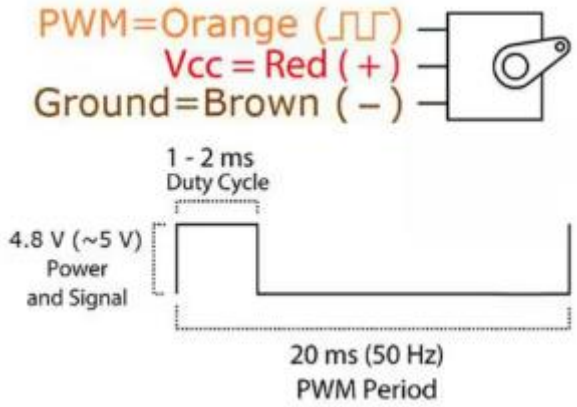
This is a non-contact ultrasonic ranging sensor of high-performance from URM09 series, employing standard PH2.0-3P Gravity vertical interface. It provides an accuracy of 1% within an effective measuring range of 2~500cm(test on a flat wall). The ranging function of this sensor is triggered by high/low-level output from the I/O ports of a host, and then the sensor converts ultrasonic flight time into a high pulse to output. The difference is that this sensor multiplexes input and output signals onto one signal port, which allows users to easily use the sensor for distance measurement only by one I/O port of a controller. Meanwhile, the adoption of advanced hardware and software design ensures excellent and reliable measuring performance. The module works well with 3.3V/5V controllers.

Points of Control Hardware:

Steering: Servo Motors take three inputs from the MCU, 5VDC, GND and a duty cycle based PWM control signal. The application of the servo inside of the CAV will rely on external power supplied by the 5VDC converter as to limit the deliverable power required from the board. This is done as the servo motor can draw a significant amount of current at stall which could overload the MCU. The GND connection will also be made to connect with the neutral line on the 5VDC regulator.

The control of the servos position will be directly linked to the duty cycle delivered by the MCU. Calibration of the servos position with respect to the PWM signal may be required as the model will require accurate and reliable steering input for its path planning.

Typically, the PWM signal is interpreted based upon its duty cycle, where a set neutral duty cycle of 50% will cause the servo position to remain at 0deg, then reduction of duty cycle to 0% or gain in duty cycle of 100% will cause the servo to rotate to its lower and upper -90 and +90deg.



Acceleration and Breaking:

Specifications:	
Brand:	Injora
Motor Model:	540 35T
Input Voltage Range:	7.2V-12V
Motor Diameter:	36mm
Shaft Length:	12mm
Shaft Diameter	3.17mm
Stall Current:	50A
No Load Current:	<1.5A
Mass:	184g
Motor Configuration:	Brushed DC

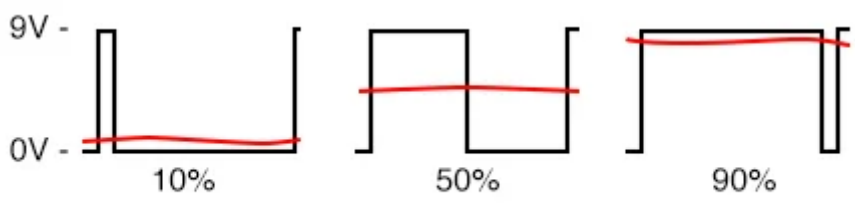
Speed Control and Breaking:

By adjusting the duty cycle of the PWM signal, you can control the average voltage applied to the DC motor. A higher duty cycle means a longer "high" time, resulting in a higher average voltage and faster motor speed. Conversely, a lower duty cycle reduces the average voltage and slows down the motor.

To control the direction of the DC motor, you need an H-bridge circuit or a motor driver. An H-bridge allows you to reverse the polarity of the voltage applied to the motor terminals. By changing the polarity and adjusting the PWM duty cycle, you can control the motor's direction and speed.

The torque produced by a DC motor is directly proportional to the current flowing through it. By controlling the average voltage with PWM, you can indirectly regulate the motor's torque. Higher average voltage results in higher current flow and more torque.

PWM signals allow for smooth transitions between different speed levels. Instead of abruptly changing the voltage supplied to the motor, you can gradually increase or decrease the duty cycle to achieve smoother acceleration, deceleration and breaking.



Examples of PWM Waveforms