4-20-2023

Implementation of Static RFID Landmarks in SLAM for Planogram Compliance

Brennan L. Drake
Georgia Southern University

Implementation of Static RFID Landmarks in SLAM for Planogram Compliance

An Honors Thesis submitted in partial fulfillment of the requirements for Honors in *Mechanical Engineering*

By
Brennan Drake
Under the mentorship of *Dr. Rocio Alba Flores*

ABSTRACT

Autonomous robotic systems are becoming increasingly prevalent in everyday life and exhibit robust solutions in a wide range of applications. They face many obstacles with the foremost of which being SLAM, or Simultaneous Localization and Mapping, that encompasses both creation of the map of an unknown environment and localization of the robot in said environment. In this experiment, researchers propose the use of RFID tags in a semi-dynamic commercial environment to provide concrete landmarks for localization and mapping in pursuit of increased locational certainty. With this obtained, the ultimate goal of the research is to construct a robotics platform for planogram compliance and inventory management to provide consistency between online retail platforms and brick and mortar stores. The platform of choice is the Turtlebot3 Burger platform, by ROBOTIS, modified to hold an RFID reader. With existing packages, researchers are provided with the ability to essentially perform SLAM on a base level using an inbuilt Lidar sensor. It is from these existing packages that researchers plan to build a system to localize RFID tags in generated maps to provide a quantifiable decrease in localization time and increase in certainty.

Thesis Mentor: *Dr. Rocio Alba Flores*

Honors Dean: Dr. Steven Engel

April 2023
Department of Mechanical / Electrical Engineering
Honors College
Georgia Southern University

Table of Contents

List of Figures

Acknowledgements

I would like to thank my mentor Dr. Rocio Alba-Flores for her support, time, and help in constructing this work, my long-time friend Austin Hightower for the opportunity and networking to be able to work on such an endeavor, and the Honors College at Georgia Southern University for a memorable and full undergraduate experience. I also want to acknowledge that this project would not have been possible without the support of my partner Peyton Munson who kept me sane amongst frustration and noisy sensor data.

1. Introduction

Autonomous Robotic systems are becoming increasingly prevalent in everyday life. From food delivery to planogram compliance, they have found themselves becoming embedded in social, practical, and logistical aspects of society. Researchers are tasked with designing systems that can process sensor information about the environment and act accordingly amongst a large magnitude of uncertainty and noise. With that in mind, a system must be designed to be either reactionary or with pseudo intelligence reliant on observation of the environment around them. Specifically in this work, along with many in this field, the question isn't exactly where the robot is from an algorithmic and control standpoint but rather, where is the robot most likely and how can this likelihood or certainty be increased? This probabilistic control methodology and study is described by the idea of Stochastic modeling[1], which entails the generation of possibilities given random variables.

While stochastic modeling and control algorithms have been around for some time, the move into modern robotics has been sparked by the emergence of new sensor technology. This age of robotic system design has been heavily impacted by the work "Probabilistic Robotics"[2] as written by Thrun, Burgard, and Fox as almost every paper referenced in development of this work were linked to it in some way. Their work synthesizes various broad topics through appeals to fundamental concepts such as the Bayes filter, Markov localization, Simultaneous Localization and Mapping, and control processes.

Probabilistic robotics accounts for uncertainty with probability distributions generally taking the form of normal or gaussian distributions. These distributions are refined through the use of complex algorithms such as scan matching to minimize uncertainty. Applying this rationale to autonomous robotic localization entails correlating sensor data with a system belief which then serves to prune unlikely positions. One such probabilistic localization method, Markov Localization (fig 1), uses sensor measurements to determine probability of all possible locations in a space. Through a process of prediction and correction steps the system is then able to refine its exact position as it traverses the modeled space.

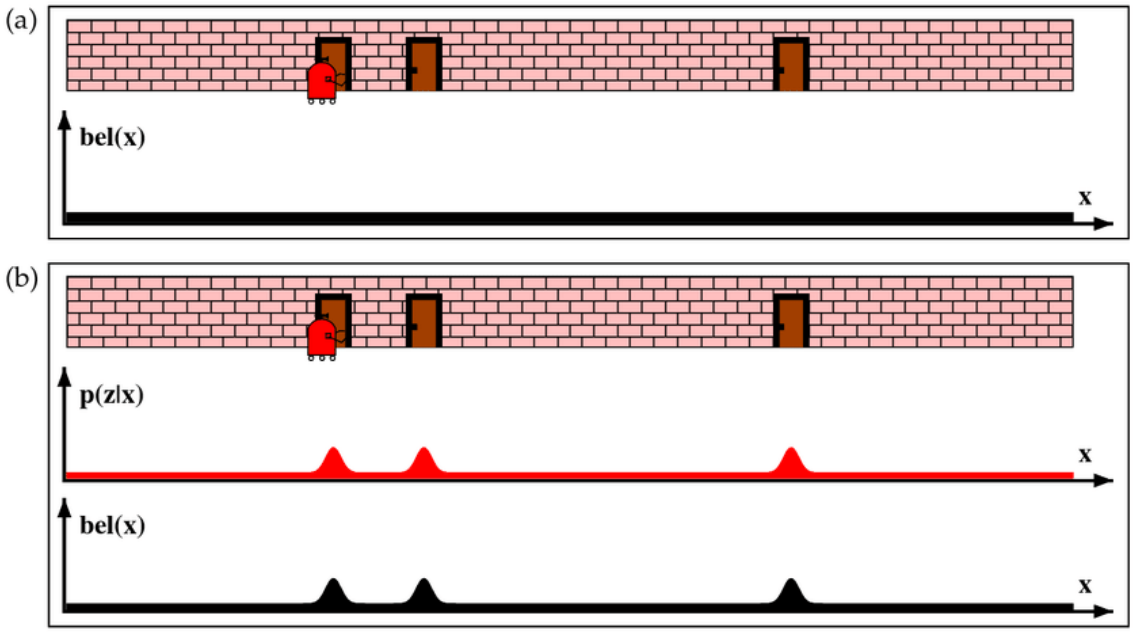


Figure 1. Markov Localization [2]

1.1 Sensor Fusion

A key aspect of autonomous robotic system construction and infrastructure is the sensor architecture by which they can obtain information about their environment. While

one sensor may prove to be viable for a slow-moving system in a static environment, multiple sensors prove to be more robust and better suited for real world applications.

Sensor fusion is a fundamental aspect of autonomous robotics as there are countless examples of the use of odometry (positional information based on motor rotation data from encoders) in conjunction with different sensors such as Light Detection and Ranging (known as Lidar), visual methods, or other novel methods of sensory data gathering in the field today. Specifically in this work, researchers aim to fuse odometry, Lidar, and Radio Frequency Identification detection events for the purposes of mapping and localization. In the case of this work, researchers will be using an “LDS-02” Lidar module which is a rotating laser rangefinder that provides the system with instantaneous distance correlated with a specific degree. By visualizing this data, a 2D point cloud can be formed that shows surrounding objects at an instance in time. This field is highly developed and so there is already a wealth of resources available for fusing odometry and Lidar data for localization.

1.2 Autonomous Robotics

In the field of autonomous robotics, there is the concept of landmarks. Landmarks are unique features whose location can be affirmed through repeated observation. In practice, with the use of sensors such as Lidar, localization algorithms must autonomously determine if an object in the environment has previously been observed (thus identifying a landmark). As certainty increases, these landmarks allow for the algorithm to model the environment around the system more accurately.

Looking at this phenomenon from another perspective, what would happen if the system were to incorrectly attribute an observation to those observed previously? The mapping and localization effort would rapidly deteriorate, yielding incorrect results. While Lidar provides a dense source of instantaneous localization information, one thing it does not provide is a method of unique identification; this must be inferred.

As seen in previous works such as that of Chen and Wang [3], RFID tags, specifically short-range passive tags, provide the vehicle for unique landmark creation and identification. In this work researchers aim to investigate the efficacy of implementing passive RFID tags into the "lifelong learning" process enabled by the "slam_toolbox" ROS2 package.

SLAM, or simultaneous localization and mapping, describes the process of using an autonomous robot to generate a map of an unknown environment while also localizing itself in that environment. At its core, the pursuit to solve this problem is fundamental to the creation of autonomous robotics systems, vehicles, and architecture in a systematic and proactive manner rather than purely reactive methods. As of now, solving the full SLAM problem, that is, estimating the entire path and location of the robot along this path, is highly based upon probability distributions. Specifically, the SLAM problem can be described by the distribution:

$$P(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \qquad (1)$$ [2]

Where x_k is the robot's pose (position and orientation) at time k, m is the full map of the environment, $Z_{0:k}$ and $U_{0:k}$ describe the measurements and controls respectively from time 0 to time k, and x_0 describes the initial pose. This distribution represents the probability distribution of estimating the position at time k and the map given sensor

measurements and control commands from time 0 to time k, and the initial position. This estimation is probabilistic due to the fact that all measurement devices have noise, or uncertainty, associated with them which compounds resulting in increasingly inaccurate and scattered maps of large environments.

With that in mind, the fundamental end goal of SLAM algorithms is to increase locational certainty over time among noisy sensor data and a potentially dynamic environment. Central to all major established SLAM algorithms is the concept of "Loop Closure"[4]. Loop closure is defined as the process of increasing certainty throughout a robot's path by returning to previously measured locations and biasing previously mapped data so that the current measured location aligns with that previously measured. Loop closure is based on recognition of features, processing of statistical expectations, and as researchers will propose, static landmarks relying on unique sensor data.

As previously stated, SLAM concerns the idea of mapping an environment and localizing oneself in that environment, but realistically in many scenarios where the use of such a system is considered, there is a degree of priori knowledge about the environment. Maybe there is an established map or a general environmental expectation such as in the focus of the proposed work where researchers claim an experimental environment of warehouses or hardware stores like Home Depot or Lowes, which have product bays and aisles. In this work, researchers are proposing to use RFID tags as landmarks to be located at each of the environment's bays for use in robot localization but also in inventory management. It is in the use of these RFID tags that researchers expect to achieve an increased certainty in robot localization, thus resulting in lower times to

accurately and confidently map an environment and ultimately position itself correctly relative to each RFID tag for planogram compliance analysis.

2. Literature Review

In regards to finding robust solutions to the full SLAM problem[5], there are many avenues of research and so in this paper, researchers have categorized a portion of them into those helpful for the current project. In this paper, these works were categorized into “Fundamental Works” which includes papers and textbooks that provide a wide breadth of information on the subject including large fundamental concepts to the SLAM problem, “Novel SLAM Approaches” that presents major research areas and applications of robust solutions to the full SLAM problem, and “Supplemental Works” that addresses isolated RFID antenna and reader construction and development. Such Novel Approaches seen in this paper include biologically inspired applications appealing to neurobiology and short term memory, Deep Learning or Machine learning applications to SLAM, and applications of RFID tagging to SLAM with robotics.

2.1 Fundamental Works

When one engages in research into the field of autonomous robotics, there are many potential directions, therefore surveying the fundamental works become essential. In the interests of this work, researchers consulted major textbooks such as the Springer Handbook of Robotics[5] and Intro to AI Robotics by MIT Press[4] as well as papers introducing fundamental architectures and concepts that current research may be based on. These major concepts specifically include early path planning or exploration

algorithms and existing fundamental filtering techniques such as the conception of Rao-Blackwellised Particle Filtering[6] .

The Springer Handbook of Robotics breaks the SLAM problem down into major dimensions through what they call the “Taxonomy of the SLAM Problem” which defines major paradigms for developing solutions to the problem. As they assert, developers must decide if their system will be volumetric or feature based in its environmental recognition, if the system will operate in a static or dynamic environment, if there is large or small uncertainty when it comes to measurements, and if the system is active or passive (does the SLAM algorithm directly impact the movement of the robot or is it teleoperated?). These are very fundamental questions that impact the architecture and function of the entire robotic system. Another source researchers consulted was Introduction to AI Robotics[4] which raises the question of where does the desired system lie on a spectrum of intelligence ranging from automation to autonomy. In doing so researchers must consider the extent that a system can generate its own plans and to which its actions are non-deterministic. They must also consider whether the system models its environment as an open world, and to the extent by which it takes inputs in the form of symbols rather than specific signals or direct commands. While these works serve to detail fundamental concepts and considerations in intelligent robotic system design, researchers look to an ever growing body of research for avenues of exploration.

One such work [7] published in the IEEE transaction on robotics, synthesizes many current research directions in pursuit of answering two main questions. The first question proposed is: “Do Autonomous robots need SLAM?”. In response, the writers assert that research into SLAM has stimulated interesting, engaging, and useful research

pertaining to sensor fusion and specialized application. They also state that SLAM's reliance on loop closure and accurate topology provides an innate defense to wrong-data association through prediction and validation. Lastly, they assert that SLAM is needed for applications requiring a globally consistent map.

2.2 Novel Approaches

The second question that readers are posed with is: "Is SLAM solved as an academic research endeavor?". Many would argue that the full SLAM problem is solved when it comes to relatively static environments and a slow moving robotic system with Lidar or visual recognition systems[7]. However, considering this to be true, it must be realized that realistic environments are not static, nor will slow moving vehicles always fit the desired use case. Because of this, there are still many avenues to pursue in research. One such avenue is to look to how landmarks are logged and recognized in the system.

Researchers at the Chinese Academy of Sciences are investigating applying mathematical, and accordingly programming, models of neurological memory to the SLAM problem. As they call it, "Shunt Memory"[8] is modeled in such a way that it exponentially decays and can be affirmed by repeated observation of the same object, place, or measurement. This is advantageous as it seems to coincide strongly with the idea of loop closure, which is relatively necessary for robust and accurate mapping and localization of robotic systems. In the same realm of the modeling of neurobiology, researchers have dubbed a method called RATSLAM[9][10] that utilizes visual data and processes it similarly to how a rat hippocampus would. DOLPHINSLAM[11] (fig 3) also applies ideas of RATSLAM to underwater 3D environments in the same ways that

BATSLAM[12] (fig 2) does using sonar waves as its medium of measurement. Each of these three explorations are prime examples of how interest in SLAM has provoked research utilizing different sensors inspired by living organisms.

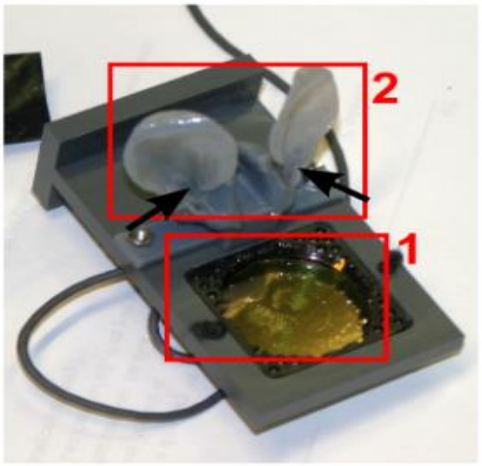


Figure 2. BatSLAM Sensor Arrangements

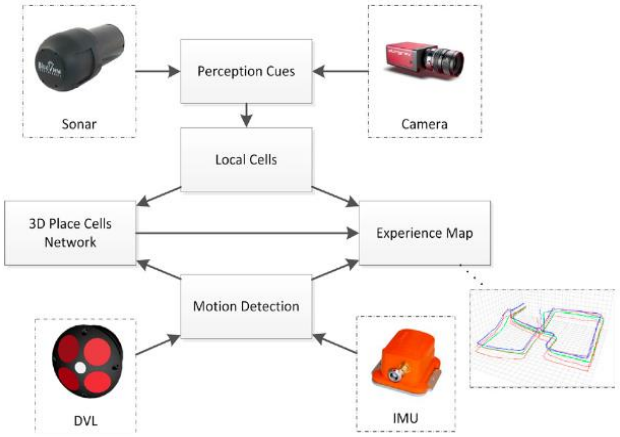


Figure 3. DolphinSLAM Control Flow Chart

While a focus on the biological and neurological processes of learning and cognition is an active and growing research area, others look to machine learning, deep learning, and reinforcement learning. Although these areas are similar and relate to biological inspirations, they heavily rely and focus on general models for learning rather than investigating the neurobiological processes of one species. One such example in this field utilizes a reinforcement learning approach with a Rao-Blackwellized Particle Filter[13] where the system can be trained in one environment and tested in another.[14] Furthermore, it is worthwhile to note that no initial map is provided to the system regardless of prior training; this map is generated online as the path planning is what is being developed through reinforcement learning.

Research into biologically inspired solutions to the SLAM problem often presents the use of different and interesting sensors, such as the use of SONAR in BATSLAM.

SLAM utilizing RFID is a research area that has some development. For instance, Chen J et al, has investigated the use of short-range passive tags and readers for use in SLAM when such tags are “sparse[ly]” placed in the environment.[3] Such a study found this practice to be feasible in the case of a passively controlled system. There are also a variety of studies using RFID for obstacle location and/or avoidance and individual localization of tags.[15][16]

2.3 Supplemental Works

In addition to RFID’s application to localization of tags and its application to robotics, there is a large amount of research into isolated RFID development. Studies such as one by Kunze S et al[17] focus on the use of RFID for distance estimation based on signal strength specifically with Ultra High Frequency RFID signals, tags, and readers. It is proven through research like this, that UHF RFID methods produce solutions to long or medium range applications. There is also research into using phase difference for distance estimation.[18]

3. Conceptual Background

3.1 The Robot Operating System

The architecture utilized in this work is the Robot Operating System (ROS) [19], which is an open source middleware allowing for the creation of nodes which

communicate through a system of publishers, subscribers, services, and actions (figs 4 and 5).

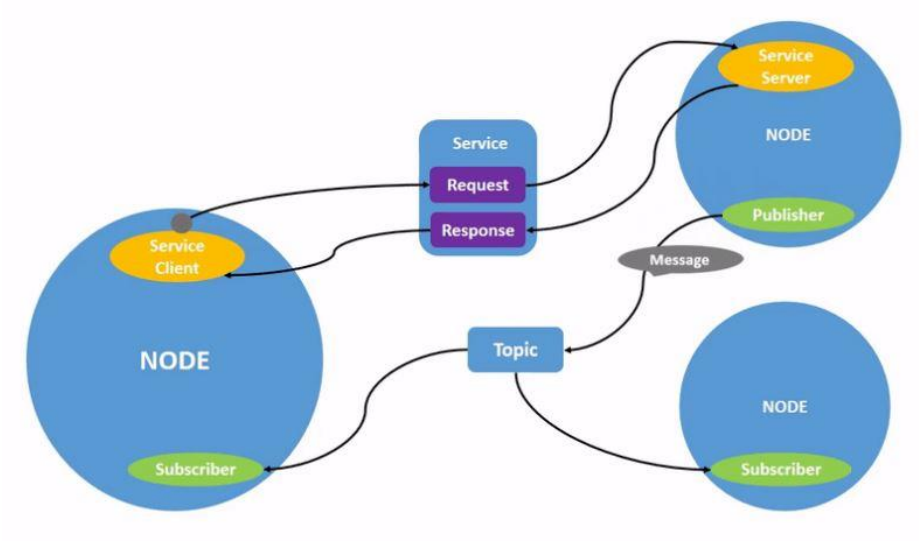


Figure 4. ROS2 Publisher Publishing Message to Topic [20]

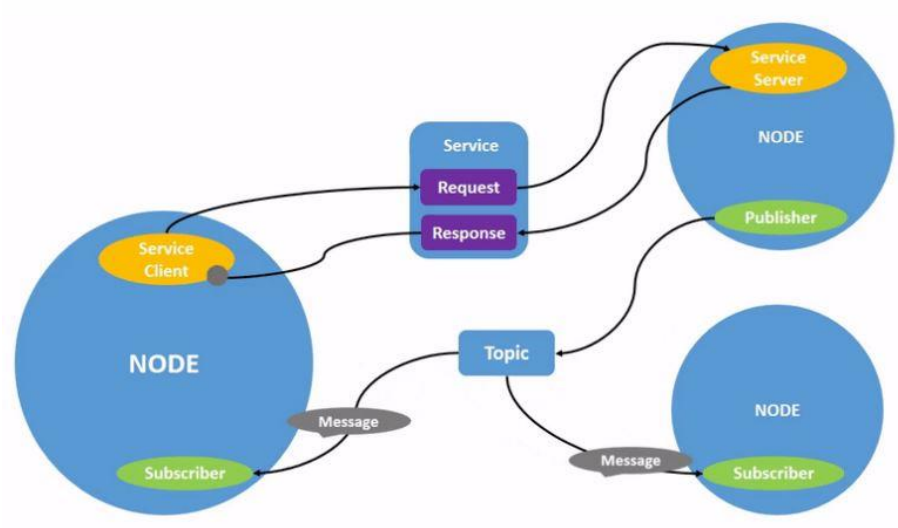


Figure 5. ROS2 Subscribers Listening to Topic for Message [20]

Specifically in this work, the proposed package was constructed in ROS2 Foxy Fitzroy [20] in conjunction with a mobile workstation running Ubuntu 20.04 Focal. ROS2 is appealing for such a project due to the potential for longevity, its open source nature, and because there is a large community and base of existing code and classes based around it. There are many existing ROS2 packages for navigation, mapping, and locomotion of the ROBOTIS Turtlebot3 Platform, the system to be used in this work, which makes initialization of the robot relatively easier.

3.2 Origin and implementation of SLAM

Simultaneous Localization and Mapping describes the process by which an autonomous robot localizes itself in an environment while simultaneously mapping out this environment. As will be explored in this work, the SLAM problem (on a basic level) has been solved, yet its application has proven to be extremely challenging due to three main factors:

1. Developed solutions to the Full SLAM problem assume a static environment and a slow-moving platform utilizing Lidar and/or visual techniques.

2. In most applications, approximating the world as static proves to limit the response capability of the autonomous system leading to undesirable behavior and inaccurate system assumptions.

3. Not all applications can accommodate a slow-moving system.

When talking about Simultaneous Localization and Mapping, researchers look to specific methods of localization for use in the system. In this work, a brief overview and exploration of application of Monte Carlo Localization (also widely known as particle filtering), occupancy Grid Mapping, and Pose Graph Optimization will be provided. Each of these major localization methods rely on a couple of distinct principles. First and foremost, the main operation and objective of each of these is to make assumptions about the environment the agent is in, and then through sensor fusion and locomotion steadily increase certainty. As can be expected, different methods prove to be attractive in different ways. Initially, researchers are looking at implementation with the "slam_toolbox" package as it provides for accurate SLAM in large and complex environments. In doing so, researchers are able to use traditional methods and existing algorithms available in open source on Github or the like to map out the environment and then inject the static RFID landmarks into this map simultaneously.

3.3 Why SLAM and why RFID?

Now we ask ourselves, why SLAM? In practice, in industry, it is common that the environment surrounding the robotic agent is not known. In addition, in many instances, operators may have a map of the environment (like a blueprint of a building) that is incomplete or inaccurate as it does not contain furnishing, moved products, or dynamic objects such as people. The world is innately dynamic and thus developing autonomous robotics that take this into account is essential for robust function. SLAM accounts for this variance.

SLAM is fundamentally based on repeated observation and the system's acknowledgement of this data association. With that being said, a major fault case for systems implementing SLAM is the wrong data association problem in which a system falsely attributes data to past observations. This failure case is most common in environments containing patterns or those lacking unique features to be treated as landmarks. Environments such as grocery stores and warehouses can prove to cause wrong data association due to reoccurring patterns of shelving. As previously mentioned in subsection 1.2, RFID technology provides a sensor by which a system could identify unique landmarks in the environment (mitigating the possibility of wrong data association).

RFID, or Radio Frequency Idenfitication, systems are a method of storing, transmitting, and receiving data via radio waves. At its most basic form, an RFID system consists of a reader and tag which both have antennas for transmitting and receiving radio waves. As a vehicle for unique identification, RFID technology is attractive because using passive tags (which are powered by incoming signals from a reader) complex electrical equipment requiring direct energy input is only required on the system reading the tags rather than on the tag in question. In addition, as opposed to barcodes, RFID tags do not require line of sight making them easier to implement into autonomous systems.

4. Method

4.1 Hardware Considerations

In this work researchers propose the use of off the shelf consumer NFC (Near Field Communication) readers, specifically PN5180 readers, as a proof of concept for the use of short-range RFID tags to improve certainty in currently available SLAM packages. With that in mind, researchers acquired a ROBOTIS Turtlebot3 and modified it to accommodate a PN5180 RFID Module and an Arduino Uno for data processing.

Researchers used the Turtlebot3 platform developed by ROBOTIS [21] as a base for proof-of-concept research. This platform, with a footprint of approximately $36in^2$ and a height of 8 inches, consists of two Dynamixel motors, an OpenCR board, a Raspberry Pi, an LDS interface board, and an LDS (Laser Distance Sensor). The Dynamixel motors contain their own gearing, network identity, encoders, drivers, and controllers. The OpenCR board serves as a control interface between the Raspberry Pi and the Dynamixel Motors while also regulating power to the system from the battery. The Raspberry Pi 4b+ with 2 GB of memory, serves as the intelligent controller of the system where it can publish sensor data and listen for control commands to execute.

In conjunction with the Turtlebot3 platform, researchers have a mobile workstation for teleoperation over a wireless network capable of running Rviz for visualization and Gazebo for simulation. This platform will be on the same network as the robot and will be where the control algorithms are executed.

4.2 Software Considerations

Looking to the software architecture for this work, researchers decided to use ROS2 Foxy in conjunction with Ubuntu 20.04 Focal and existing SLAM packages as noted in subsection 3.1.

The ultimate research plan is to initialize the Turtlebot platform to perform established SLAM algorithms which will then be modified to use RFID detection events to create landmarks in the environment. Researchers will then simulate the use of these landmarks in algorithms using Gazebo visualized in Rviz , physically modify the Turtlebot platform with RFID readers, and then create a test environment with RFID tags to use for SLAM.

One of the first obstacles that must be tackled in this project is to develop the interface between the Turtlebot3's Raspberry Pi and the Arduino in terms of the serial connection and how it relates to ROS2. This can be broken down into three main goals which will be explored in the following section:

1. Ensure the Arduino runs firmware correctly for the PN5180 reader and condense the detection data into a format easily used.
2. Alter this firmware to send the data via serial port to the Raspberry Pi (and ensure it's reception through the Linux "/dev" directory.
3. Write a ROS2 node to read the data from the incoming serial port and publish this detection data, entailing a detection event and RFID tag ID to a topic "/rfid".

With that accomplished, researchers then face the task of implementing the detection events into the "slam_toolbox" package as developed by Steve M.. [22].

5. Experimental Initialization

5.1 Simulation Considerations

Gazebo and Rviz, mentioned earlier, are two powerful tools for use in autonomous vehicle control algorithm development. Gazebo is a simulator with the ability to simulate physics and accurate platform models, while Rviz is a tool to visualize the data that the system is taking in and how it is processing that data. The main difference is that Gazebo is a tool for simulating the environment of the system and Rviz is a tool for displaying the robot's interpretation of this environment. Gazebo will prove invaluable in that researchers can simulate the placement of RFID tags along with the proposed RFID reader configuration on the robot and how it affects the use of various control algorithms without having to physically construct the system.

5.1.1 Robot State Publisher and the Unified Robot Description Format (URDF)

The "Robot State Publisher" is a ROS node that, when initialized, continually publishes the transforms or relations between different coordinate frames on the system. For instance, with the Turtlebot3, the "base_scan" frame where the Lidar measurements are in relation to, is offset from the robot's "base_frame". Similarly, the frames of the two wheels are offset via a transform from the "base_frame" as well. These offsets are declared in a "URDF" file (according to the robot's physical dimensions) and processed by the robot state publisher node.

A "URDF" file is a markup file that describes the relations between all frames on the robot. It can also be used to establish collision geometry for simulation, inertia, as

well as sensor types. In this work, researchers modified the URDF file provided by ROBOTIS for the Turtlebot3 in order to add the RFID assembly. The main utility of this is twofold: 1. It allowed them to ensure simulations account for the possible collision of this front assembly, and 2. It gave them a location for the RFID detection events to be originating from for use in visualization through Rviz.

As seen in the figures below (figs. 6, 7), this alteration consisted of the addition of an “RFID_base_link” which references the stl files of the structure to be explained in subsection 5.2 and an “RFID_sensor_link” which is precisely where the sensor data detection ray would originate. In figure 7. note how the “RFID_sensor_link” is relative to the “RFID_base_link” which is relative to the “base_link”. In robot URDF design, it is best practice to have all frames be ultimately in reference to a base_link such as this (seen in Appendix A1 as parent and child declarations).

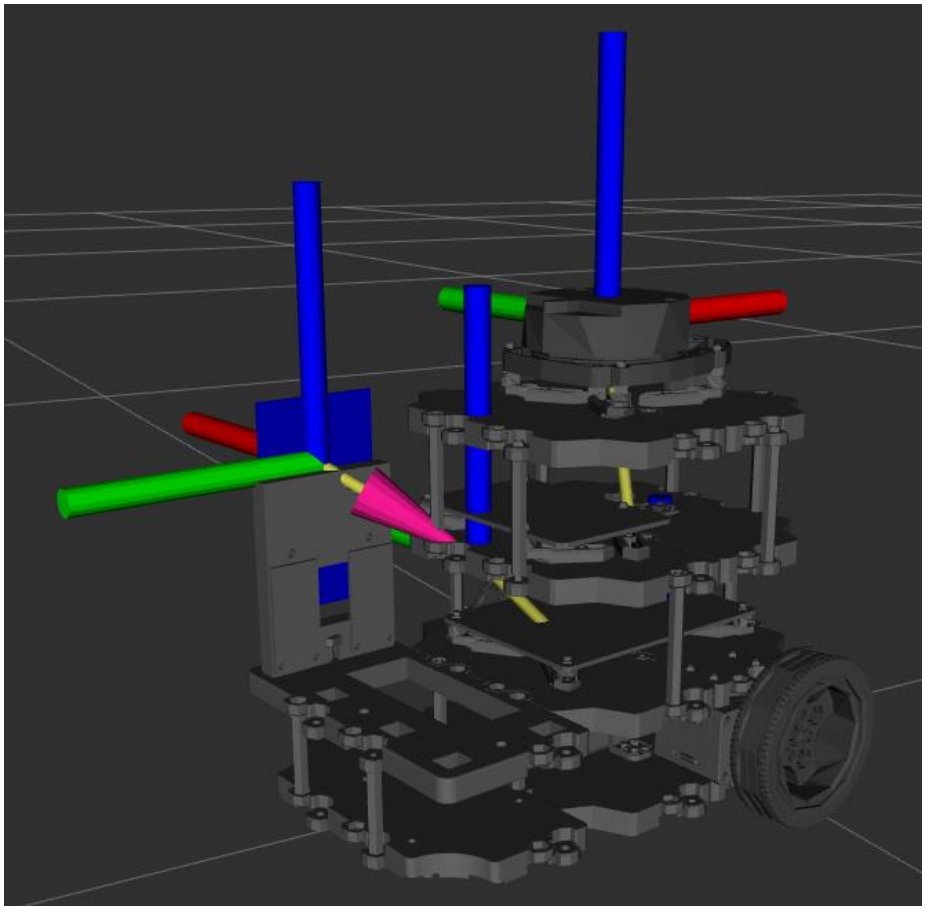


Figure 6. Rviz2 Simulation Assembly with Models

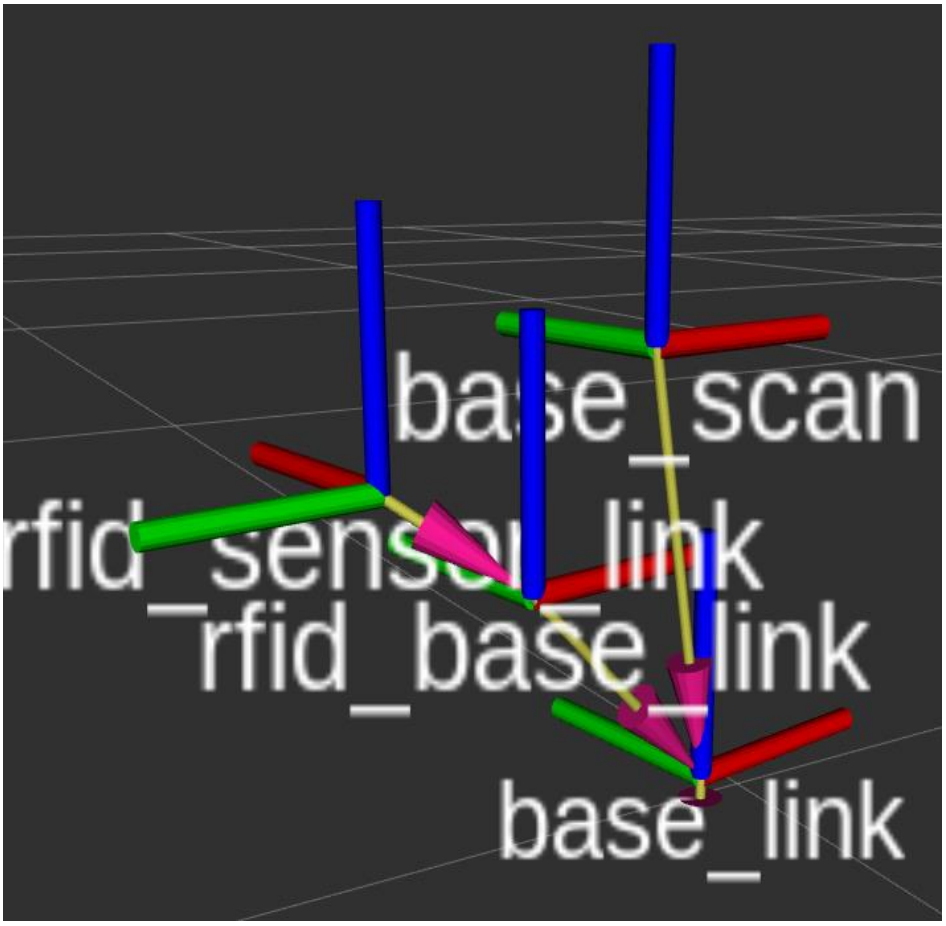


Figure 7. Rviz2 Frame Visualization

5.2 Structural Construction

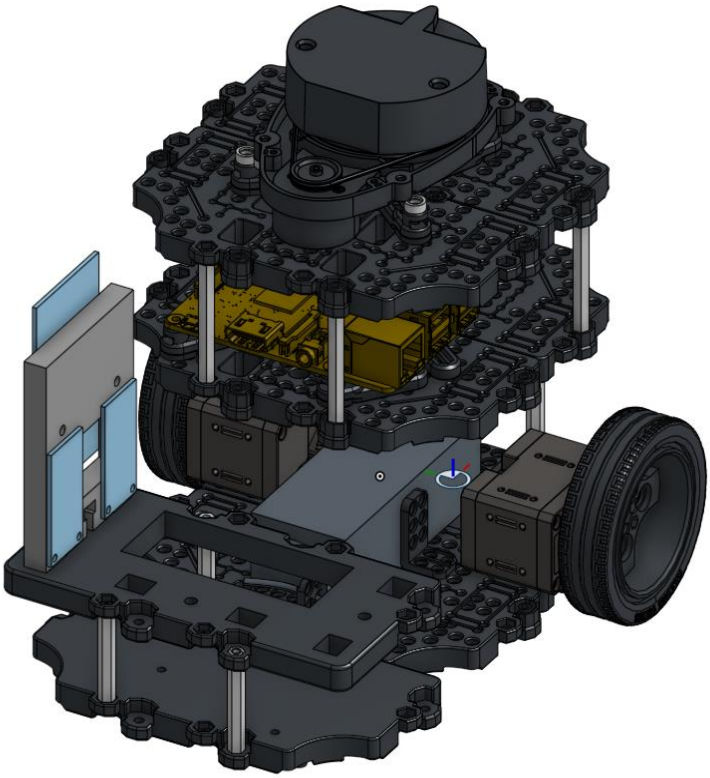


Figure 8. Turtlebot3 with RFID Assembly

The construction of the Turtlebot3, manufactured by ROBOTIS, is open source. This means that all of the 3D models of the individual components are available online on OnShape for anyone to access for free.[23] In this work, researchers accessed these files and modified the baseplate file to accommodate an Arduino Uno, and then again in order to be able to mount the PN5180 readers and prototype boards (seen in figure 9. and file links seen in Appendix A2). This design was created in order to accommodate a maximum of three PN5180 readers, but only one was deemed necessary for the purposes of this work. The structure was manufactured on an Ender3 FDM 3D printer using PLA with a cubic infill of 50%.

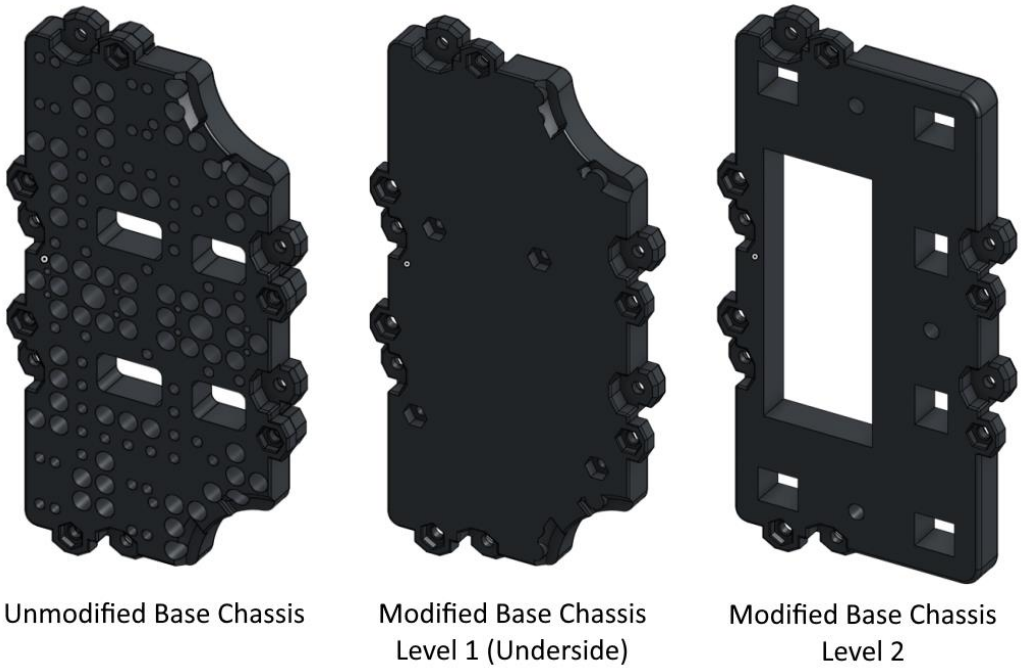


Figure 9. Modified Turtlebot3 Base Plate in OnShape

As previously mentioned, researchers modified the baseplate to accommodate PN5180 readers. The specific design took use of a double slot system (fig 10) in which the assembly is rigidified by a constrained nut. This design was inspired by a fastening system seen in many consumer robotics kits and other assemblies using laser-cut plywood.

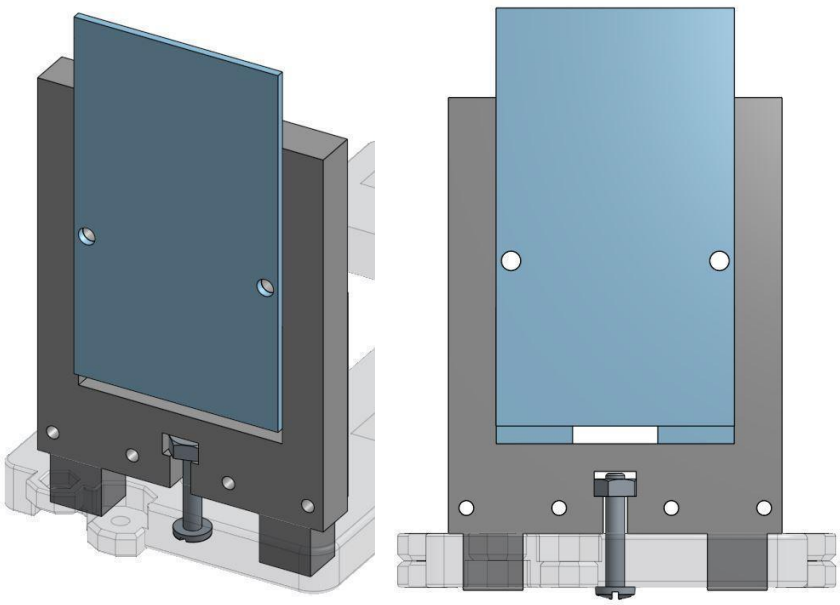


Figure 10. PN5180 Mounting System

5.3 Circuit Construction

In this work, researchers utilized four main components for the RFID reader assembly. Researchers used a PN5180 for RFID tag recognition, logic level shifters, an Arduino Uno for signal processing and serial output for the main control system, and a glass fast blow fuse for overall circuit protection and redundancy. The wiring in this work was done primarily with prototyping perfboard, solid core 22awg copper wire, and screw terminals. In the following sections, each major component aside from the fuse will be explored in detail.

PN5180 readers utilize a Serial Peripheral Interface or SPI interface for communication between a master device, in this case the Arduino microcontroller, and

the reader, also known as the slave device. The basic construction of SPI communication consists of three main terminals, a MOSI terminal (Master In Slave Out), a MISO terminal (Master Out Slave In), and an SCK or system clock terminal that ensures the two devices are synchronized in their input and output. In addition, the PN5180 reader has a 5V power terminal, a 3.3V power terminal, a ground terminal, a BUSY terminal, a system select terminal, and a RESET terminal. In operation the reader takes in commands into the MOSI terminal (as it is the slave device) and outputs information out of the MISO terminal. When a specific reader is operating on commands given to it, the BUSY line will be high, the NSS line is used to signal to a reader that it is the device selected which is used in signal processing, and the RESET terminal when pulled high RESETs the reader allowing it to reinitialize.

In this experiment, researchers used an Arduino Uno for processing of the RFID data from the PN5180 reader. Due to existing libraries primarily developed by ATRAPPMAN [24] and Playful Technologies [25], implementation of such a system was relatively trivial when it came to actual software implementation. With that in mind, researchers did face another hurdle that the logic level shifters proved to solve. Arduino Uno operate with a logic of 5V. In layman's terms, this means that when an output GPIO pin on the Arduino Uno is set to HIGH it outputs a signal of 5V. In contrast, the PN5180 readers operate on a logic level of 3.3V and so to use a logic level of 5V would easily damage these sensitive components.

With that in mind researchers implemented simple I2C logic level shifters in which a reference voltage on either side was provided along with corresponding signal lines. This proved to be a viable solution. Figure 11. shows a reference wiring diagram

for the signal level shifting and the diagram of this experimental circuit. Another reference diagram by Playful Technologies is seen in Appendix A3 as well.

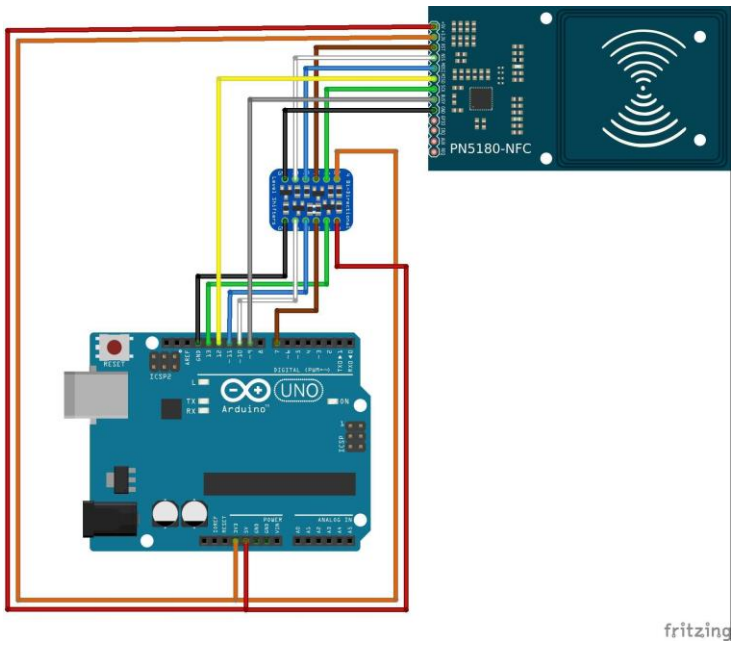


Figure 11. ATRAPPMAN Wiring Diagram

5.4 Codebase and Software Design

This work is focused around creating a package for use in ROS2 Foxy Fitzroy due to the wide availability of existing SLAM algorithm packages available today. While developing the physical hardware, researchers developed a flow chart of the ROS architecture to be created (figure 12.). This architecture includes the mechanism by which the RFID data is published to a ROS2 topic and then how it is accessed by the rest of the system.

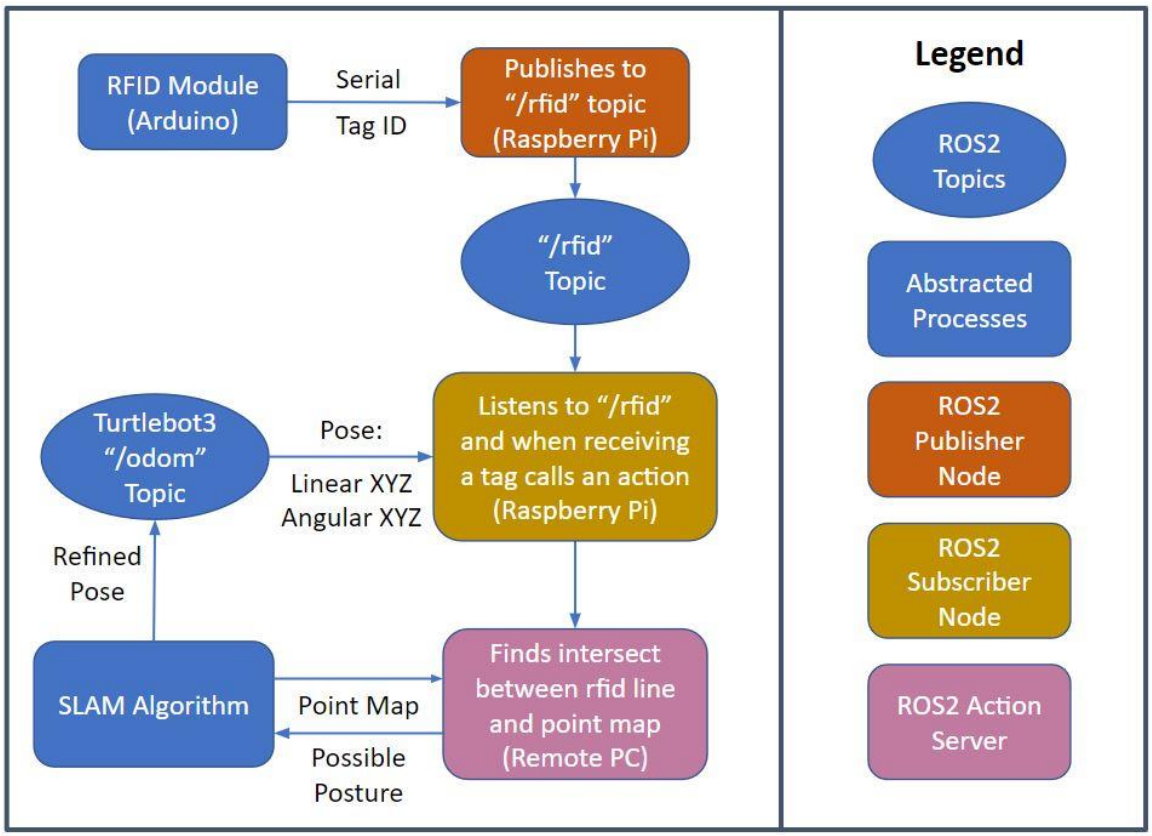


Figure 12. Control Architecture Flow Chart

The first element drafted for this work was a system by which the Raspberry Pi, also seen as the brains of this robot, would take in the serial RFID data from the Arduino and provide it to the rest of the system. The implementation developed by researchers was to create a publisher that would continuously be listening to this RFID data stream on the TTYACM0 Serial Port and publish that data to the topic "/rfid" using a String message from the std_msgs packages provided in the base ROS installation.

As mentioned previously, there have been many SLAM algorithms developed in the past, though only two major methods dominate ROS2. Cartographer and the SLAM Toolbox. In this work, researchers aim to modify the SLAM Toolbox for our purposes.

The SLAM Toolbox, developed by Steve Macenski [22], is a ROS2 package for SLAM that takes features from KartoSLAM and Cartographer (entailing graph-based operation) to optimize mapping for larger spaces with a mobile processor.

6. Implementation

6.1 Physical Construction

For the purposes of this work, researchers printed the structural components, consisting of the two plates, and PN5180 mount out of blue PLA with 50% infill on an Ender 3 FDM printer. The final files mentioned earlier were developed out of a need to simplify the models given by the manufacturer for ease of printing. Overall, the printing process was simple and straightforward. With a fit tolerance of 0.2 millimeters the parts interfaced as expected and physical assembly was completed.

In constructing the circuit, researchers looked primarily at the schematic as seen in figure 11 by ATRAPPMAN. The differences, reflected in figures 13 and 14, were mainly to allow the Arduino and reader assembly to be powered from the battery of the Turtlebot3, to include a fuse for safety, and to allow for more sensors to be added in the future. With that in mind, observers will notice the use of two logic level shifters. If adding more readers, all would share one shifter for their I2C pins (MOSI, MISO, and SCK) and use another shifter for each reader for the data pins individually (NSS, BUSY, and RST). The construction of this circuit used solid and stranded core 20 gauge wire. It is important to note that the wires may not loop or curve significantly directly behind the reader as this will cause interference (resulting in what looks like failing firmware).

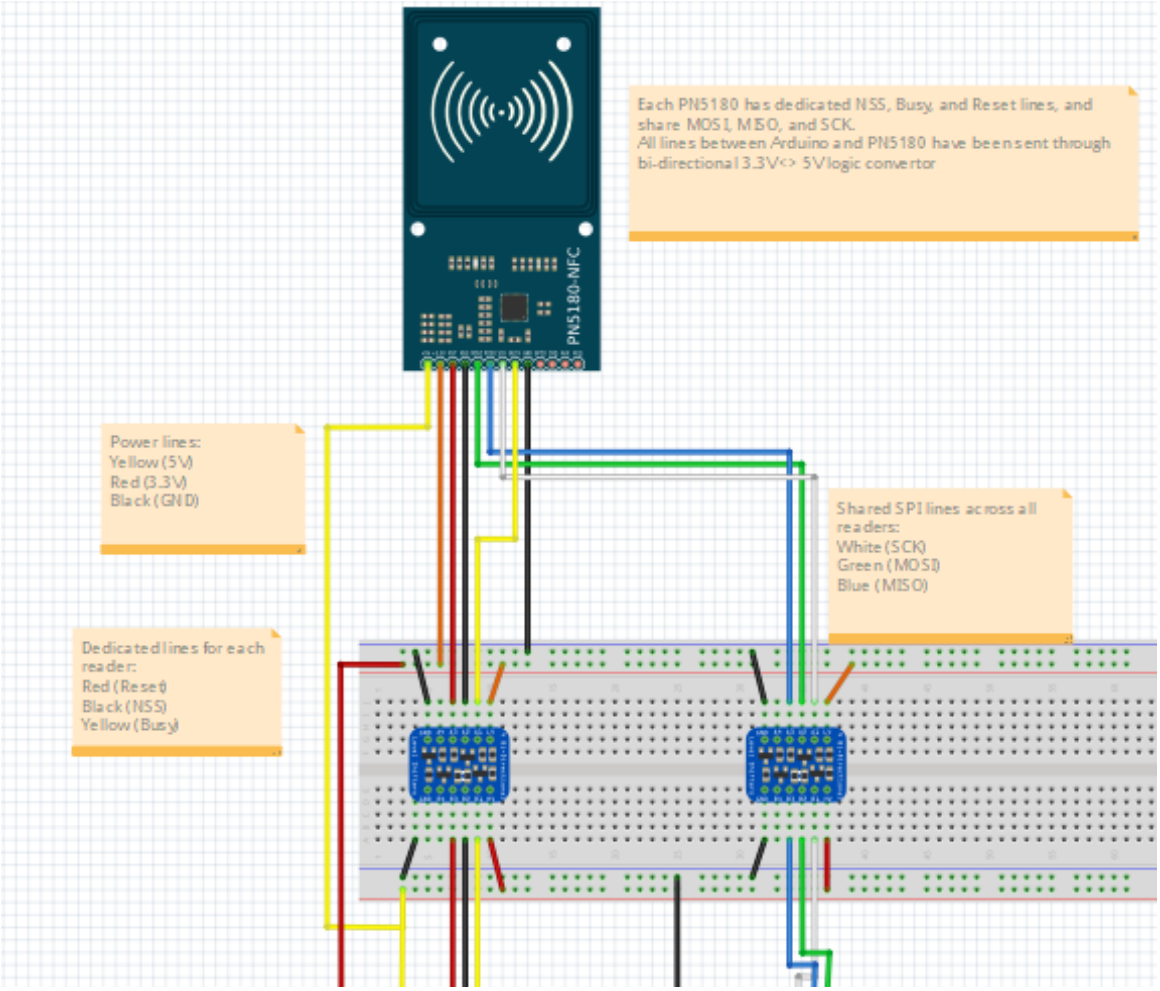


Figure 13. Circuit Fritzing Part 1

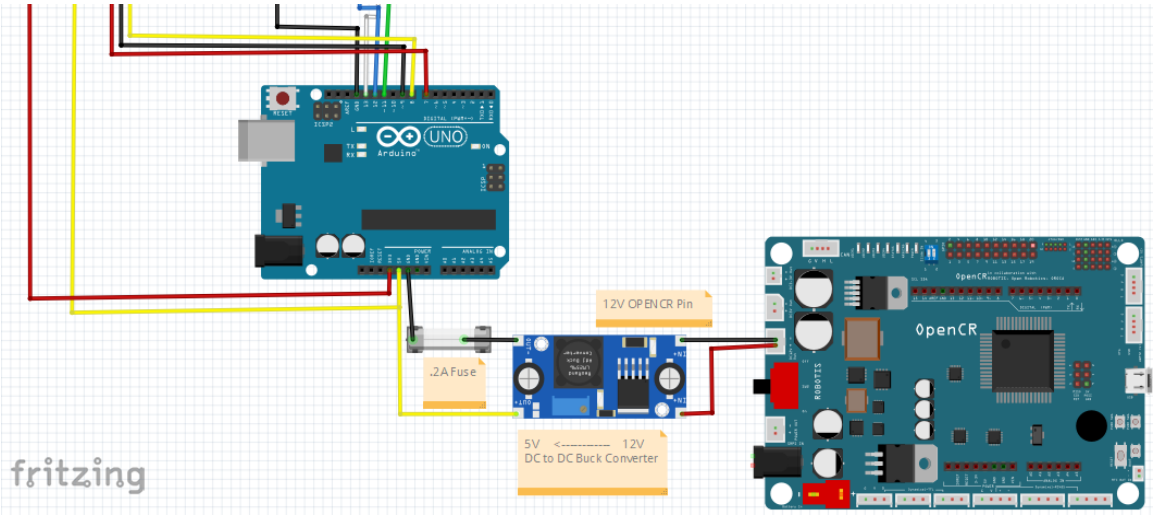


Figure 14. Circuit Fritzing Part 2

In allowing the system to be powered from the battery, researchers note that the OPENCR board, which powers the Raspberry Pi and other devices, has a 5V pin (where the pi draws power) and a free 12V pin. In testing, researchers attempted to power the reader assembly and the pi from the 5V pin in parallel but found that the increased load led to a voltage drop undervolting all components. To avert this, researchers used a buck converter and stepped the 12V pin down to 5V and powered both the lower side of the logic level shifter, the PN5180 reader, and the Arduino from this source. Note that a 0.2 Amp glass blow fuse was also put in this connection to ensure protection against catastrophic battery failure and OPENCR overload.

Moving on to look at the Arduino connections. In this circuit there are 6 main data wires (fig 15, 16). MOSI, MISO, and SCK are for the SPI data connection with the PN5180 and are shared among all readers. BUSY, NSS, and RESET are then also determined to provide certain functionality with the reader (such as resetting it, selecting which reader to read from, and to affirm that a specific reader is BUSY and cannot be

sent command). In constructing the circuit, researchers used the color code of MOSI, MISO, and SCK being green, blue, and white wires respectively. In addition, red, yellow, and black align with BUSY, NSS, and RESET respectively (note that these are specifically for signal wires and red, yellow, and black also refer to 3.3V, 5V and ground wires in the system. This repetition of colors is due to material on hand though signal and power wires are easy enough to differentiate between).

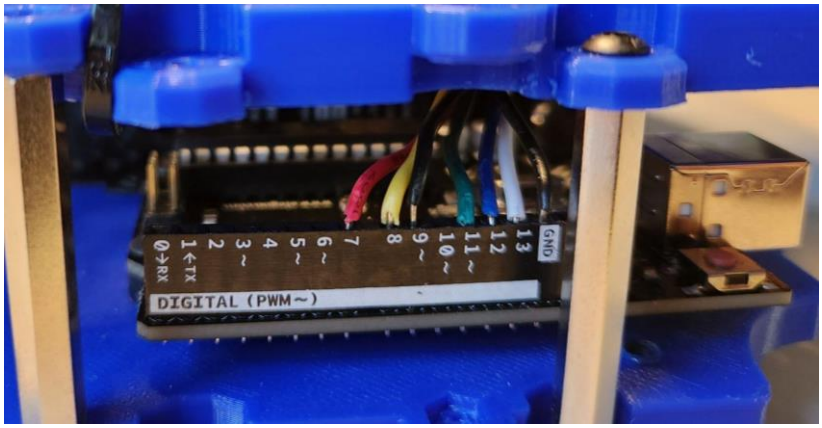


Figure 15. Arduino Pin Locations

```
79    #define PN5180_NSS  8
80    #define PN5180_BUSY 7
81    #define PN5180_RST  9
```

Figure 16. Arduino Firmware Pin Declarations

6.2 Software Construction

When looking to interface PN5180 NFC readers with Arduino or similar microcontrollers, firmware provided by the user “ATRAPPMAN” on Github proves to be essential. This firmware handles identification and initialization of the readers along the

SPI interface. For the current purposes of this work, researchers forked this code and edited it to output the RFID Tag ID continually at a rate of 4 times per second. This firmware already outputs log information to the serial monitor, so the modifications were minor and are reflected in the repository provided (Commented out sections are also reflected in Figure 17. below). With this groundwork laid, the RFID tag information is being provided to the serial port of the Raspberry Pi.

```
157  void loop() {
158    if (errorFlag) {
159      uint32_t irqStatus = nfc.getIRQStatus();
160      //showIRQStatus(irqStatus);
161
162      if (0 == (RX_SOF_DET_IRQ_STAT & irqStatus)) { // no card detected
163       // Serial.println();
164       // Serial.println((empty));
165      }
166      nfc.reset();
167      nfc.setupRF();
168      errorFlag = false;
169    }
170
171    //Serial.println(F("-----------------------------------"));
172    //Serial.print(F("\nLoop #"));
173    //Serial.println(loopCnt++);
174    Serial.println();
175    delay(250);
176
177    uint8_t uid[8];
178    ISO15693ErrorCode rc = nfc.getInventory(uid);
179    //Serial.print(F("Inventory successful, UID="));
180    for (int i=0; i<8; i++) {
181      Serial.print(uid[7-i], HEX); // LSB is first
182      if (i < 2) Serial.print(":");
183    }
184  }
```

Figure 17. Arduino Looping Code

The next step was then to write code to run on the Raspberry Pi that would read this serial data and publish it to a ROS topic (fig 18). The main reasoning for this is that the RFID detection event and ID would then be available to control software anywhere in the ROS2 Foxy network such as the ultimate end goal, SLAM architecture. The basis of this code would be a simple publisher script (or “talker” script) as detailed in ROS2

documentation [26] that would listen to the ttyUSB0 port (where the Arduino is plugged in) and publish the RFID data to the "/rfid" topic.

```python
#! /usr/bin/python3

import serial
import rclpy
from rclpy.node import Node
from std_msgs.msg import String


class MinimalPublisher(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'rfid', 10)
        timer_period = 0.05  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0
        self.ser = serial.Serial("/dev/ttyUSB0", 115200, timeout=1)

    def timer_callback(self):
        msg = String()
        if ser.in_waiting > 0:
            id = ser.readline().decode('utf-8').rstrip()
        msg.data = id
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1


def main(args=None):
    rclpy.init(args=args)

    minimal_publisher = MinimalPublisher()

    rclpy.spin(minimal_publisher)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_publisher.destroy_node()
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

Figure 18. "/rfid" Topic Publisher Code

This script was then added to the turtlebot3_bringup launch file which initializes the robot for control. This addition is seen in the following figure where the package

containing the code in figure 19. is in the “rfid_publisher” package and is set to be identified as “rfid_talker” in the setup.py file.

```
95          Node(
96              package="rfid_publisher",
97              executable="rfid_talker",
98              output="screen"),
99
100     ])
```

Figure 19. Bringup Launch File Alteration

As an initial proof of concept, researchers constructed a script that would subscribe to the “/rfid” topic and either rotate or move forward for one second depending on which tag was held in front of the reader. This code is seen in Appendix A4 and serves as a solid benchmark for reception and processing of the RFID data. From here, researchers were tasked with formulating a method to visualize these detection events in RVIZ and implement them into existing SLAM algorithms.

7. Conclusions

In closing, researchers have established the experimental bench and completed all but the control algorithm development. Through the work presented so far, the PN5180 reader has proven to be a viable tool for short-range RFID detection and, when paired with robotics architecture such as the Robot Operating System, a useful sensor.

More specifically, researchers have modified a Turtlebot3 platform to accommodate an RFID reader assembly consisting of a single PN5180 reader, an Arduino Uno running firmware for the PN5180 reader, logic level shifters, and a glass

blow fuse for fire-protection. Working with ROS2 Foxy Fitzroy, they have also created a script to be included in the “turtlebot3_bringup” package that makes the RFID detection data available to the entire ROS2 Network. Having made the RFID data available for control system use, researchers then developed test code to demonstrate robotic identification of unique tag IDs.

Moving forward, researchers are looking to where they can modify the SLAM Toolbox to increase locational certainty when a tag is observed. As seen previously in figure 12., the next components of this system architecture would be to develop:

1. A ROS2 node that would look at the current estimated odometry (position and angular posture of the system in the environment), and when receiving an RFID detection event, project a line from the current position 15cm into the environment where the tag was detected. This node would then call an action server with the parameters being the unique tag ID and this line (referred to as the RFID Line) broken into discrete points.

2. The aforementioned ROS2 Action server, which would identify the intersection between the point map constructed by the SLAM algorithm and the input “RFID Line”. This intersect would then be attributed to the unique identity of the RFID tag (with an appropriate covariance surrounding).

3. Modifications to the SLAM_Toolbox that reinforce the system’s belief of a position when reading a unique tag identity that has been previously observed.

Works Cited

[1] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*. Academic, 1979.

[2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics. The MIT Press, Cambridge, Massachusetts, 2005.

[3] Chen, J. F., & Wang, C. C. (2010). Simultaneous localization and mapping using a short-range passive RFID reader with sparse tags in large environments. Proceedings of IEEE Workshop on Advanced Robotics and Its Social Impacts, ARSO, 136–141. https://doi.org/10.1109/ARSO.2010.5680012

[4] Murphy, Robin R. "15: Localization, Mapping, and Exploration." Essay. In Introduction to AI Robotics, 2nd ed. Cambridge, MA: The MIT Press, 2019.

[5] Raja Chatila et al. "Part E Moving in the Environment": Springer Handbook of Robotics. Springer Nature. Berlin (2016): pp. 1133-1381.

[6]Doucet, A. (2000). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. UNCERTAINTY IN ARTIFICIAL INTELLIGENCE PROCEEDINGS.

[7] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., & Leonard, J. J. (2016). Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Sensor Age. IEEE Transactions on Robotics, 32(6), 1309-1332. https://doi.org/10.1109/TRO.2016.2624754

[8] Li, Y., Li, S., & Ge, Y. (2013). A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments. Neurocomputing, 104, 170–179. https://doi.org/10.1016/J.NEUCOM.2012.10.011

[9] Milford, M. J., Wyeth, G. F., & Prasser, D. (2004). RatSLAM: A hippocampal model for simultaneous localization and mapping. Proceedings - IEEE International Conference on Robotics and Automation, 2004(1), 403–408. https://doi.org/10.1109/ROBOT.2004.1307183

[10] Yu, S., Wu, J., Xu, H., Sun, R., & Sun, L. (2020). Robustness Improvement of Visual Templates Matching Based on Frequency-Tuned Model in RatSLAM. Frontiers in Neurorobotics, 14, 68.

[11] Silveira, L., Guth, F., Drews, P., Ballester, P., Machado, M., Codevilla, F., Duarte-Filho, N., & Botelho, S. (2015). An Open-source Bio-inspired Solution to Underwater SLAM. IFAC-PapersOnLine, 48(2), 212–217. https://doi.org/10.1016/J.IFACOL.2015.06.035

[12] Steckel, J., Peremans, H., & Ouzounis, C. A. (2013). BatSLAM: Simultaneous Localization and Mapping Using Biomimetic Sonar. Citation: Steckel J, Peremans H, 8(1), 54076. https://doi.org/10.1371/journal.pone.0054076

[13] Doucet, A. (2000). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. UNCERTAINTY IN ARTIFICIAL INTELLIGENCE PROCEEDINGS.

[14] Botteghi, N., Sirmacek, B., Ai, J., Mustafa, K. A. A., Poel, M., & Stramigioli, S. (2020). On Reward Shaping for Mobile Robot Navigation: A Reinforcement Learning and SLAM Based Approach. https://doi.org/10.48550/arxiv.2002.04109

[15] Lemus, R., Díaz, S., Gutiérrez, C., Rodríguez, D., & Escobar, F. (2014). SLAM-R Algorithm of Simultaneous Localization and Mapping Using RFID for Obstacle Location and Recognition. Journal of Applied Research and Technology, 12(3), 551–559. https://doi.org/10.1016/S1665-6423(14)71634-7

[16] Forster, C., Sabatta, D., Siegwart, R., & Scaramuzza, D. (2013). RFID-based hybrid metric-topological SLAM for GPS-denied environments. Proceedings - IEEE International Conference on Robotics and Automation, 5228–5234. https://doi.org/10.1109/ICRA.2013.6631324

[17] Kunze, S., Poschl, R., & Grzemba, A. (2015). Signal strength based distance estimation of passive off-the-shelf RFID tags in the UHF band. 2015 1st URSI Atlantic Radio Science Conference, URSI AT-RASC 2015. https://doi.org/10.1109/URSI-AT-RASC.2015.7302992

[18] Povalac, A., & Sebesta, J. (2011). Phase difference of arrival distance estimation for RFID tags in frequency domain. 2011 IEEE International Conference on RFID-Technologies and Applications, RFID-TA 2011, 188–193. https://doi.org/10.1109/RFID-TA.2011.6068636

[19] "Robot operating system," ROS. [Online]. Available: https://www.ros.org/. [Accessed: 18-Jan-2023].

[20] "Foxy Fitzroy." *Foxy Fitzroy (Foxy) - ROS 2 Documentation: Foxy Documentation*, Open Robotics, 2023, https://docs.ros.org/en/foxy/Releases/Release-Foxy-Fitzroy.html.

[21] "AI Research Starts Hereros Official Platform." ROBOTIS.US, ROBOTIS, https://www.robotis.us/turtlebot-3/.

[22] SteveMacenski. "Stevemacenski/slam_toolbox at Foxy-Devel." Github, 22 Feb. 2022, https://Github.com/SteveMacenski/slam_toolbox/tree/foxy-devel.

[23] "ROBOTIS Turtlebot3 Chasis CAD Files Onshape," Onshape. [Online]. Available: https://cad.onshape.com/documents/2586c4659ef3e7078e91168b/w/14abf4cb615429a14a2732cc/e/9ae9841864e78c02c4966c5e. [Accessed: 18-Jan-2023].

[24] ATrappmann. "Atrappmann/PN5180-Library: PN5180 Library for Arduino." *Github*, 19 Aug. 2021, https://Github.com/ATrappmann/PN5180-Library.

[25] Playfultechnology. "Playfultechnology/Arduino-RFID-PN5180: Interfacing Arduino with 13.56MHz ISO15693 RFID Tags Using PN5180 Modules." *Github*, 14 Dec. 2021, https://Github.com/playfultechnology/Arduino-RFID-PN5180.

[26] “Writing a Simple Publisher and Subscriber (Python).” *Writing a Simple Publisher and Subscriber (Python) - ROS 2 Documentation: Foxy Documentation*, Open Robotics / ROS Community, https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html.

Appendix

A1 Turtlebot3 URDF File Modification

```
78    <joint name="rfid_base_joint" type="fixed">
79      <parent link="base_link"/>
80      <child link="rfid_base_link"/>
81      <origin xyz="-0.085 0.0125 0.087" rpy="0 0 0"/>
82    </joint>
83
84    <link name="rfid_base_link">
85      <visual>
86        <origin xyz="0 0 0" rpy="0 0 3.14159"/>
87        <geometry>
88          <mesh filename="package://turtlebot3_description/meshes/bases/rfid_base.stl" scale="0.001 0.001 0.001"/>
89        </geometry>
90        <material name="light_black"/>
91      </visual>
92
93      <collision>
94        <origin xyz="0 0 0." rpy="0 0 0"/>
95        <geometry>
96          <box size="0.140 0.140 0.143"/>
97        </geometry>
98      </collision>
99    </link>
100
101   <joint name="rfid_sensor_joint" type="fixed">
102     <parent link="rfid_base_link"/>
103     <child link="rfid_sensor_link"/>
104     <origin xyz="-0.0425 0.045 .035" rpy="0 0 1.57079632679"/>
105   </joint>
106
107   <link name="rfid_sensor_link">
108     <visual>
109       <origin xyz="0 0 -.015" rpy="0 0 0"/>
110       <geometry>
111         <box size="0.001 0.05 0.08"/>
112       </geometry>
113       <material name="blue"/>
114     </visual>
115   </link>
```

A2 Turtlebot3 CAD Files

Pyo Great contributor Mar '17

[TurtleBot3 Hardware: Free for YOU!]

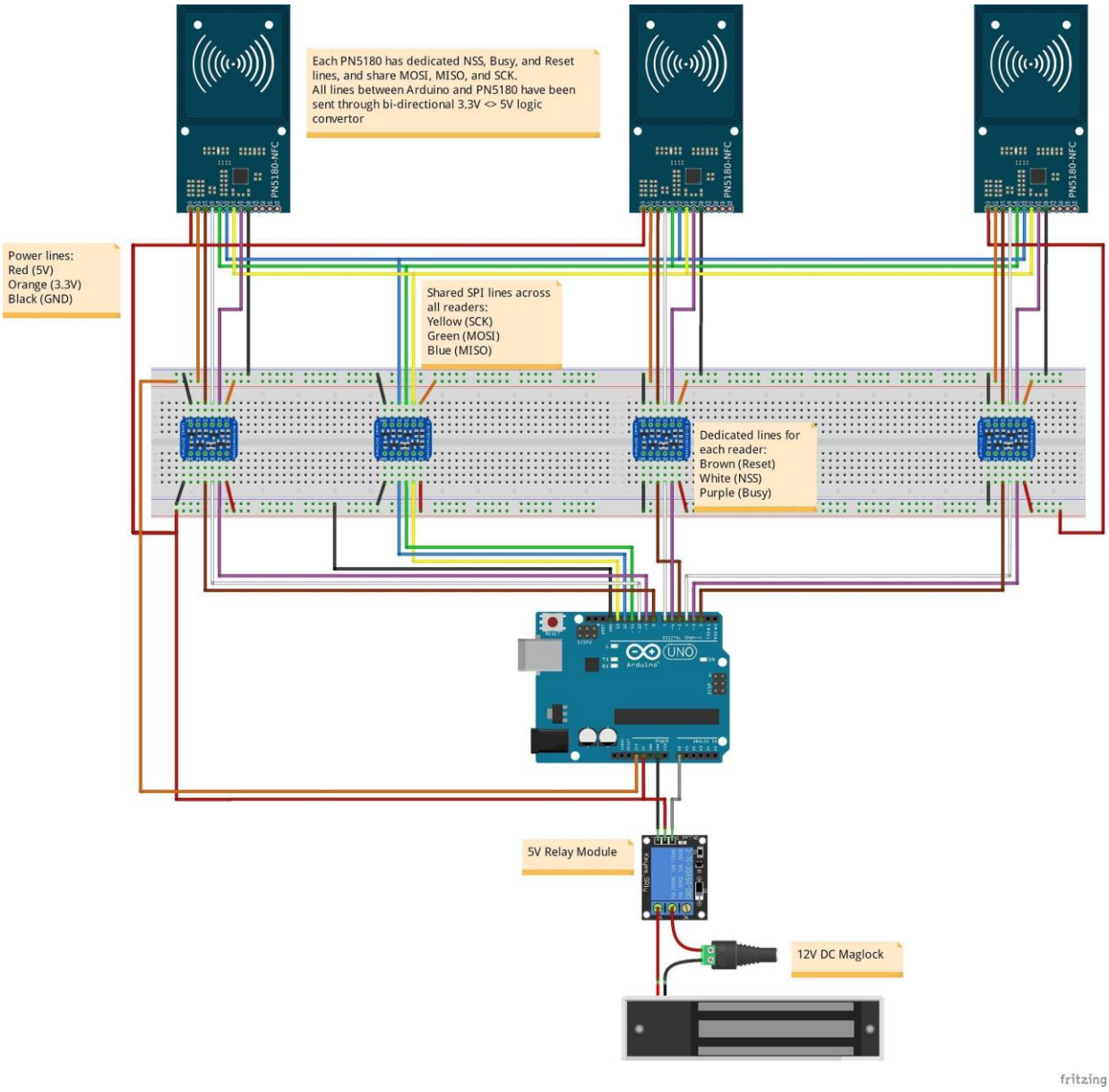
Hi, All 🙂

We are very pleased to announce the TurtleBot3 hardware through Onshape 91. You can look deeply into TurtleBot3 hardware configurations via Onshape, download STL files for 3D printing, copy design drawings, and even re-edit them. All this is done in a web browser. Oh! I forgot. You can do it on any devices and operating systems, such as your tablet, smartphone. Check it out now. 😎

BURGER:
https://cad.onshape.com/documents/2586c4659ef3e7078e91168b/w/14abf4cb615429a14a2732cc/e/9ae9841864e78c02c4966c5e 750

WAFFLE:
https://cad.onshape.com/documents/daf80b2021249a203e938429/w/417587bf5e713dc3f4985a96/e/7d4ea1bdda0c24388701b0e4 469

A3 PlayfulTechnologies Wiring Diagram



A4 RFID Test Code

```python
import time
import rospy
from std_msgs.msg import String
from geometry_msgs.msg import Twist

class Rfid_Test():
    def __init__(self):
        self.pub = rospy.Publisher("/cmd_vel", Twist, queue_size=1)
        self.cmd = Twist()
        self.rate = rospy.Rate(10)
        self.rfid_sub = rospy.Subscriber("/rfid", String, self.rfid_handler)
        self.tag_id = ""
        self.ctrl_c = False

    def shutdownhook(self):
        self.cmd.angular.z = 0
        self.cmd.linear.x = 0
        self.ctrl_c = True

    def rfid_handler(self, data):
        self.tag_id = data.data

    def control(self):
        while not rospy.is_shutdown():
            if self.tag_id == "00:0:000000000":
                self.cmd.linear.x = 0
                self.cmd.angular.z = 0
                self.pub.publish(self.cmd)
            elif self.tag_id == "E0:4:10888CE126":
                self.cmd.linear.x = 0.5
                self.cmd.angular.z = 0
                self.pub.publish(self.cmd)
                time.sleep(1)
            elif self.tag_id == "E0:4:10888CA1D7":
                self.cmd.linear.x = 0
                self.cmd.angular.z = 0.5
                self.pub.publish(self.cmd)
                time.sleep(1)
            else:
                rospy.loginfo("Initializing")


if __name__ == "__main__":
    rospy.init_node("rfid_control_node")
    rfid_robo_obj = Rfid_Test()
    rfid_robo_obj.control()
```