Computer-Sensor System for Autonomous Indoor Transport

Sonar-Infrared Beacon Sensing Augmented by Ultrasonic Shield

Benjamin Goldstein

Advisor: Professor Roman Kuc

Submitted to the faculty of the Department of Electrical Engineering in partial fulfillment of the requirements for the degree of Bachelor of Science

Department of Electrical Engineering
Yale University

May 10, 2023

[Revised December 21, 2023]

Abstract

Autonomous vehicle technology is experiencing a boom this decade: the industry has an estimated total market capitalization of over $100 billion as of 2022, and is projected to exceed $1.8 trillion by 2030 (pre 2023). Outdoor autonomous vehicle systems commonly utilize satellite-based GPS technology for coarse positioning in conjunction with LiDAR (light detection and ranging), UWB (ultra-wideband), and CV (computer vision) systems for object detection and path planning. Indoor localization, though, remains a more open issue– common personal device location software typically uses GPS with a resolution of roughly 3 meters (gps 2020). Recent developments in UWB technology have helped increase resolution at shorter ranges, although at an often restrictive pricepoint (Guo et al. 2022). In this project the group investigated a different sensor-based approach: positioning of sonar-infrared enabled beacons around the landscape for triangulation-based localization. Beginning with a legacy configuration consisting of a PC and Arduino mounted on a wheelchair chassis, the group refactored the setup into a smart, sensor-integrated system powered by a Raspberry Pi PC as well as an Arduino microcontroller for offboard signal processing. The process culminated in a proof-of-concept in which the robot was able to orient itself toward a beacon and drive until the beacon reaches a minimum distance, at which point the robot stops. In addition, the proof of concept included a PWM (pulse width modulation) based protocol for beacon identity encoding in infrared signals, paving the way for implementation of triangulation based localization in future work.

Table of contents

Acknowledgments

There are many people I'd like to thank for the success of this project. First and foremost, Professor Roman Kuc and Kevin Ryan provided a degree of technical expertise and mentorship that were integral in pushing this project over the finish line. From debugging electronics with an oscilloscope to ensuring that the right parts were always in the lab on time, these two brought constant stability and direction to a project that often desperately needed it.

I'd also like to thank my partner on the drive system team, Jonathon Durand, for the long hours spent together in the lab tinkering with sonar circuits and I/O pinouts. Without his efforts this robot would likely have never driven! I absolutely must also shout out the rest of the group: Yu Jun Shen and Austin Zhu designed a clever infrared system, providing a previously blind robot with eyes, while Sachi Sharp, Maggy Lambo, and TJ Patel created a PCB-based high power sonar system with the potential to boost the robot's capabilities to the next level by pushing the bleeding edge of range and beam-width.

Though less directly involved in the project, I'd like to thank my housemates: Reese Johnson, Adam Marcelo, Wilson Nesbit, Aidan Sze, and Cavan Walsh. They've made our residence a hospitable place to the all-nighters and engineering jargon that's characterized this project, and that would likely outright frighten most outsiders to the field. I commend their resilience, and would be remiss not to express my gratitude in this section.

I'd also of course like to thank my family and friends who are too numerous to name, but include my parents Lynne and Stephen, and number one supporter Niki Francis. Without their love and encouragement I wouldn't be here today!

Chapter 1

Introduction

The goal of this project was to investigate sensor-based, primarily infrared and sonar, approaches to automated transport. The project was structured as a redesign and augmentation of an existing system: an electronic wheelchair chassis. The wheelchair was originally designed to be operated solely by human control through a joystick apparatus, particularly by the elderly. Collisions with both pets and grandchildren were commonplace, and need for action was apparent. With this in mind, we set out to create a sensor system that augments the functionality of the joystick controlled wheelchair while keeping the safety and autonomy of the user in mind. This chapter summarizes background information pertaining to the project, the project specification, and the starting system on top of which the project was built.

1.1 Background

The problem of autonomous transport is not particularly new, and a vast corpus of existing literature along with a comprehensive suite of tools exist to solve it. Many existing technologies are powerful for globally scoped use cases, but have insignificant resolution for indoor applications. Certain higher resolution systems fail to achieve sufficient range, and others

are prohibitively costly. The design outlined in this project occupies a productive niche in the aforementioned landscape, and that should become apparent after this section.

1.1.1 Definitions

The establishment of a common vocabulary is integral to the coherence of this report. The subsection will cover various localization and positioning techniques as well as relevant signal processing terminology. GPS refers to Global Positioning System, a satellite-based radio navigation system owned by the US government and operated by the US Space Force. Radio waves refer to the portion of the electromagnetic spectrum ranging from roughly 1 Hz to 3000 GHz– GPS uses several frequency bands in this range all between 1000 and 1600 Mhz (GIS 2023). GPS is an extremely powerful system that drives the vast majority of moderate to low resolution location software, such as a smartphone's location services, up to a resolution of roughly 3 meters (gps 2020). Resolution in this context refers to the uncertainty bound up to which we can rely upon the accuracy of our reading. For example, a GPS position is only valid up to $\pm$ 3 meters in any direction.

Lidar refers to light detection and ranging, radar radio detection and ranging, and sonar sonic navigation and ranging. All three technologies utilize the emission of a signal (Lidar and radar utilize electromagnetic transmissions, while sonar deploys sound waves) and the analysis of a reflected response to determine features about the surrounding landscape. The most common such analysis is ToF (time of flight) analysis, which determines the range to an object by recording the time between transmission and detection of a reflected signal by the following relation:

$$d = \frac{v \cdot t_f}{2} \tag{1.1}$$

where d is the measured distance, v the signal velocity, and t_f the recorded time of flight. The speed of light and sound are both well known, so the time of flight is sufficient to determine range under the assumption that the signal travels in a straight line in both directions.

Many times this is not the case– due to the topology of the environment signals may reflect at strange angles producing convoluted flight paths and thus inaccurate ToF ranges. This is an open problem with this technology, and often measurements are simply taken with a high frequency and smoothed to avoid drastic effects of such anomalous measurements. Due to the limitations on the speed of low-price electronics, it should be noted that sonar systems (as well as ultrasonic systems that are analogous but with a slightly higher operating frequency) are significantly more compatible with ToF ranging.

Finally the definition subsection closes with coverage of IR (infrared), UWB (ultra-wideband), and relevant signal processing terms. Infrared refers to the electromagnetic spectrum ranging from roughly 300 GHz to 400 THz. This is the portion of the spectrum that is lower frequency but immediately adjacent to the visible spectrum. Infrared sensing is commonly used to augment visual spectrum techniques, as in the case of night vision and heat-seaking weaponry (Harder et al. 2020). Perhaps less militant applications are also worth mentioning: infrared emitter/sensor pairs are commonly used for simple remote control systems including non-internet television remotes. UWB is a novel radio technology that utilizes a wider bandwidth, encoding information in varied energy levels at sharply defined time intervals. UWB is commonly used for ranging applications as it achieves a uniquely high resolution at low to moderate ranges with ToF analysis. For this reason, UWB has been gradually integrated into leading smartphones over the past 5 years (Flueratoru et al. 2021). Digital signals refer to signals that are interpreted as discrete values, generally speaking by using a threshold. Analog signals, conversely, are signals interpreted as taking values on a continuous range. PWM (pulse-width modulation) refers to a technique that encodes information in the duty cycle of a periodic (repeating) digital signal. The duty cycle of a signal is the percentage of time that it spends as a digital 1 as opposed to a digital 0. PWM is commonly used to encode analog information (that takes values on a continuous range) in a digital signal (which takes values 0 or 1).

1.1.2 Related Work

As was eluded to in the prior subsection, ultra-wideband has emerged as the dominant ranging and localization technology in terms of both resolution and range (Flueratoru et al. 2021). It has not, however, yet experienced the price plummet that often accompanies the maturation of an emergent development into an established component of modern systems. Hence, given the use case of the project, UWB was ruled out as an option. In a similar light, much development has occurred in the field of machine-learning based image processing, which can be leveraged for pose estimation and therefore positioning (Phon-Amnuaisuk et al. 2022). The bottleneck here is compute: in order to feasibly execute the computation required to maintain accurate position data from the camera, the robot would either demand onboard compute capacity at least at the level of a modern laptop (which is prohibitively expensive and bulky to mount) or a persistent low-latency network connection for communication with a provisioned cloud cluster. The project team decided collectively that reliance on such a connection could compromise the reliability and even safety of the project, opting to primarily rely on sensing to aid the robot's movement. The relative economics of more established sensing equipment makes looking to an older result extremely compelling: sensor fusion of IR and sonar. Usage of sonar and IR in tandem to support path planning is an established method, and can be readily augmented through the use of beacons (Flynn 1988). Through the placement of beacons around the landscape, Flynn's design can be boosted to achieve higher accuracy and reliability. This realization served as the jumping off point for the project.

1.2 Legacy System

This section describes the initial system from which the team began our development. This preliminary setup can be divided into three layers: the motor driver/actuator layer, the virtual joystick/microcontroller layer (control layer 1), and the PC layer (control layer 2).

Notably absent is any sensor integration– the wheelchair chassis was a purely manually controlled entity prior to this project.

1.2.1 Motor Driver/Actuator Layer

In this subsection we discuss the physical layer, or motor drive. It is of the utmost importance to understand the system at lower levels of abstraction before layering on complexities. The wheelchair chassis is driven by two independently controlled DC motors, one driving each rear wheel. The front two caster wheels are merely to stabilize the vehicle, and are free to rotate on their vertical axis to facilitate steering. The rear wheels are fixed in their orientation. The chassis is powered by a pair of 12 V wheelchair batteries, which are typically rated on the order of 30-35 AH. The unit is also fitted with an onboard DC to AC inverter, which is not used to drive the motors but rather to power any additional systems that demand AC power. The batteries are rechargeable via a standard 120V 60Hz AC wall outlet, and are resilient to repeated charge/discharge cycles over relatively long lifespans (2 years of consistent use).

The motor drivers are supplied with two inputs per wheel: a PWM signal that governs the rotational speed, as well as an H-bridge digital input that controls direction. The underlying circuit can be diagrammed something like this:

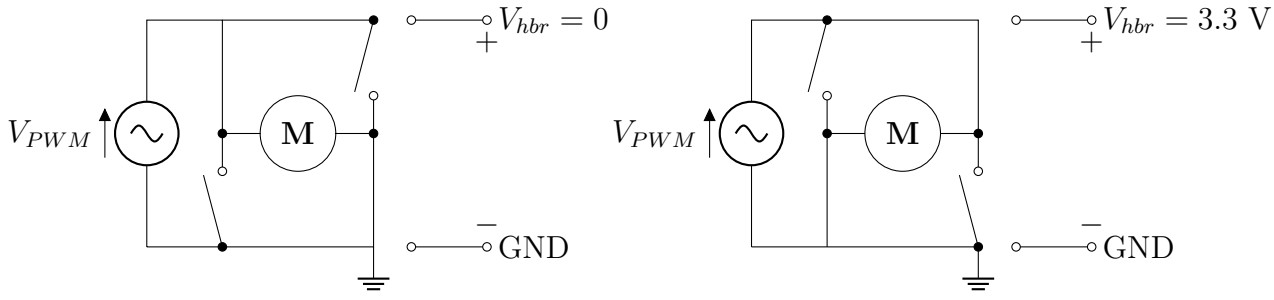

Figure 1.1: Circuit Schematic of Motor Driver Interface

Reversing the polarity of the H bridge control voltage changes the direction across which the input voltage is applied to the motor driver by toggling two pairs of switches. Under

the hood the H-bridge is likely implemented using CMOS or a comparable semiconductor technology, but the mechanical switch model is suitable for this level of analysis. The PWM polarity is always the same– only the duty cycle is altered. The exact relationship between duty cycle and rotational speed was not investigated beyond the basic understanding that higher duty cycle yields higher rotation speed. The PWM frequency is also held constant; in the legacy system it was set to roughly 32 kHz. Note that in this structure the actuator layer does not set the PWM or H-bridge control inputs, but rather exposes them as an input to the next layer.

1.2.2 Virtual Joystick/Microcontroller Layer

This layer's primary task is to translate higher level directives from the user into the voltage inputs described in the previous subsection: a PWM magnitude and digital direction (fed into the H-bridge) for each wheel. The legacy system was designed around virtualizing the physical joystick used to drive the chassis in its former role as a manually operated wheelchair. The virtual joystick was implemented with an Arduino Uno microcontroller wired as follows: Both the input select and emergency stop are powered by an on-chip pullup voltage through

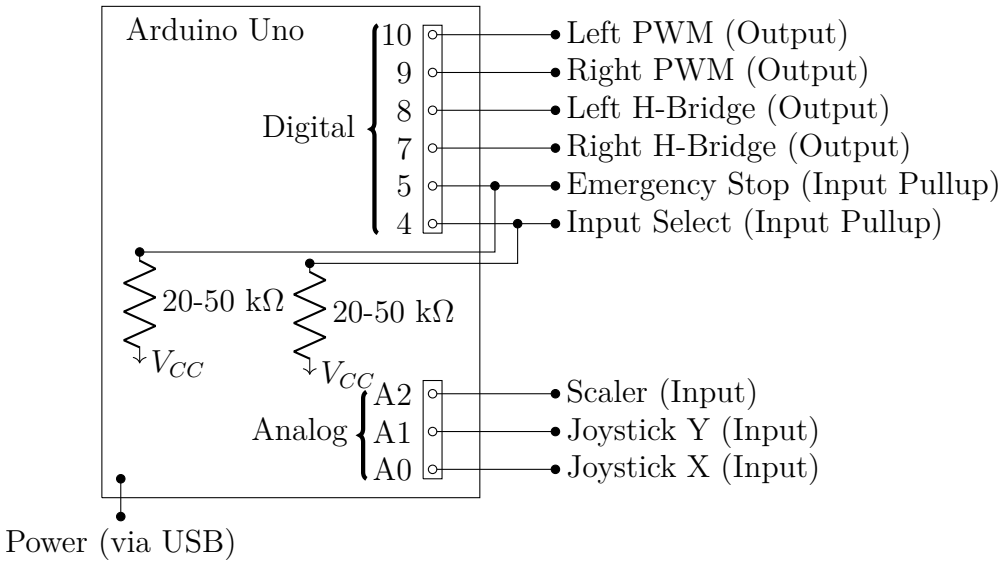

Figure 1.2: Arduino UNO Pinout of Legacy Virtual Joystick

a builtin resistor. The emergency stop and input select switches are both connected to ground in typical pullup input configuration. The emergency stop switch does exactly that in software: when pressed the robot ceases to send voltage to the motors through the PWM pins. The input select pin toggles between taking input from the physical joystick (via analog pins A1 and A2) and from the COM port (which also powers the device). In software, the Arduino computes appropriate digital values for each H-bridge and PWM output pin from the appropriate input. If the input select switch is closed (the digital read will be high), then the Arduino uses analog voltage values from the joystick taken from potentiometers whose values are altered by the physical position of the controller. Those values are then scaled by a factor derived from the value of a third potentiometer attached to a knob for global speed control. The horizontal (X) value of the joystick is associated with a rotation rate, while the vertical (Y) value of the joystick is associated with a forward/backward speed. Only after establishing forward/rotation speeds are the individual wheel speeds (and thus PWM duty cycles/H-bridge polarities) calculated. This decomposition of motion into a radial (forward) and rotational component is extremely prevalent throughout the rest of the drive system design. If the switch is closed then the Arduino polls the USB port awaiting byte-encoded speed information, again decomposed into rotational and translational terms. The virtual joystick does have the capability to serve as the top-level control unit through the physical controller, but also exposes a serial port interface for PC-based control.

1.2.3 PC/Top-Level Control Layer

Despite the direct control capability of the joystick, the primary means of control of the legacy system was through a discrete time MATLAB program. The program presented a GUI with four options: up, down, left, or right. Upon pressing the appropriate button, the robot would either step forward, backward or rotate a fixed amount left or right. The default step duration was set to 0.5 seconds, so difficulties in precise control are fairly foreseeable. MATLAB also is required to run on a desktop computer, so a Windows desktop along with

a workstation consisting of a keyboard, mouse, and monitor on a rolling cart adjacent to the chassis was required to control the drive system.

1.2.4 Legacy Integration

The section will finish by discussing the integration between the previously described layers, including all interfaces exposed between layers as well as those exposed to the end-user.

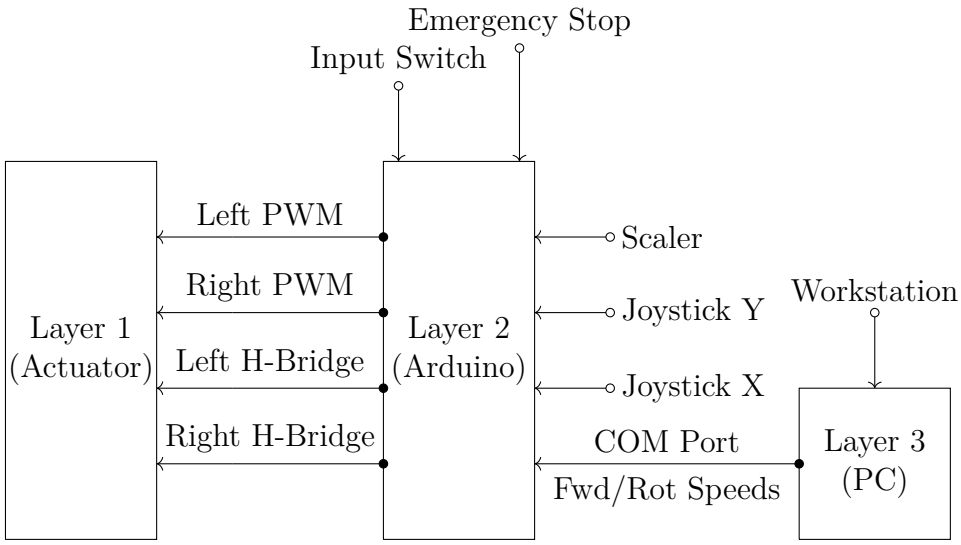

Figure 1.3: Interlayer Integration Details of Legacy System

Despite the flaws in this design, it is absolutely well designed in the sense of layered abstraction. The data flows in one direction from a high degree of abstraction in the matlab program to a low degree as voltage values to the motor driver. The main focuses after an initial observation of this system were definitely to (1) remove the dependence on a rolling workstation adjacent to the robot, (2) refactor software out of MATLAB and into a faster, more lightweight langauge, and (3) switch control from discrete time to "continuous" time (or closer to it).

1.3 IR-Sonar Circuit Design Concept

In addition to the drive system described in section 1.2, the group started off with some sense of a design for IR-sonar fusion based beacon sensing. The design consists of two separate circuits: one with a sonar emitter and IR receivera nd the other vice versa. The goal is to be able to "singaround", i.e. have the IR emitter chirp, the sonar emitter hear it and chirp back (in sonar this time) and have the IR emitter perform some type of analysis based on the nature of the received IR response. This requires not only careful circuit design, but also a knowledge of networked systems.

1.3.1 Ranging Protocol

In addition to the circuit itself, this design also introduces a novel ranging protocol based on sonar ToF analysis compared to an IR baseline. The timing diagram is seen below:

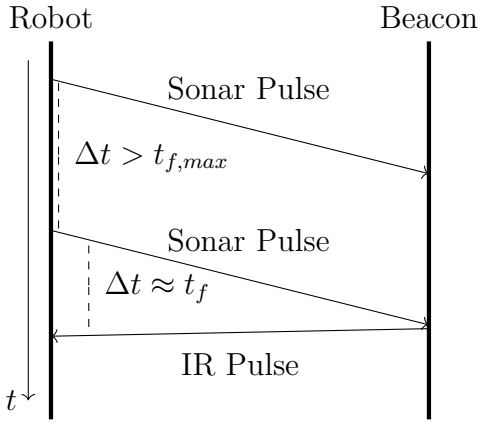

Figure 1.4: Ranging Protocol Timing Diagram

As can be seen in the timing diagram, the protocol unfolds something like this: The robot emits a periodic sonar pulse, waiting for an infrared response. If the robot doesn't get the repsonse within a reasonable amount of time, then it resends the sonar pulse. If it does get the IR response, then it records the amount of time between the sonar emission and the IR reponse, treating that as its time of flight at the speed of sound. This analysis makes

two main assumptions: first that the IR transmission occurs effectively instantly. If the IR transmission took non-negligible time then the c would have to be subtracted from the speed of sound in each calculation, which it is not. The second is that the beacon responds to the sonar hit immediately, or at the very least with low latency relative to the true time of flight. Both of these assumptions are generally true enough to rely on for ToF analysis without loss of precision, and the robot can calculate its range to the beacon via equation 1.1.

1.3.2 Circuit Design

Initially, the implementation of the protocol utilized two microcontrollers with analog sensor circuit integrations. A proof-of-concept schematic is included below:

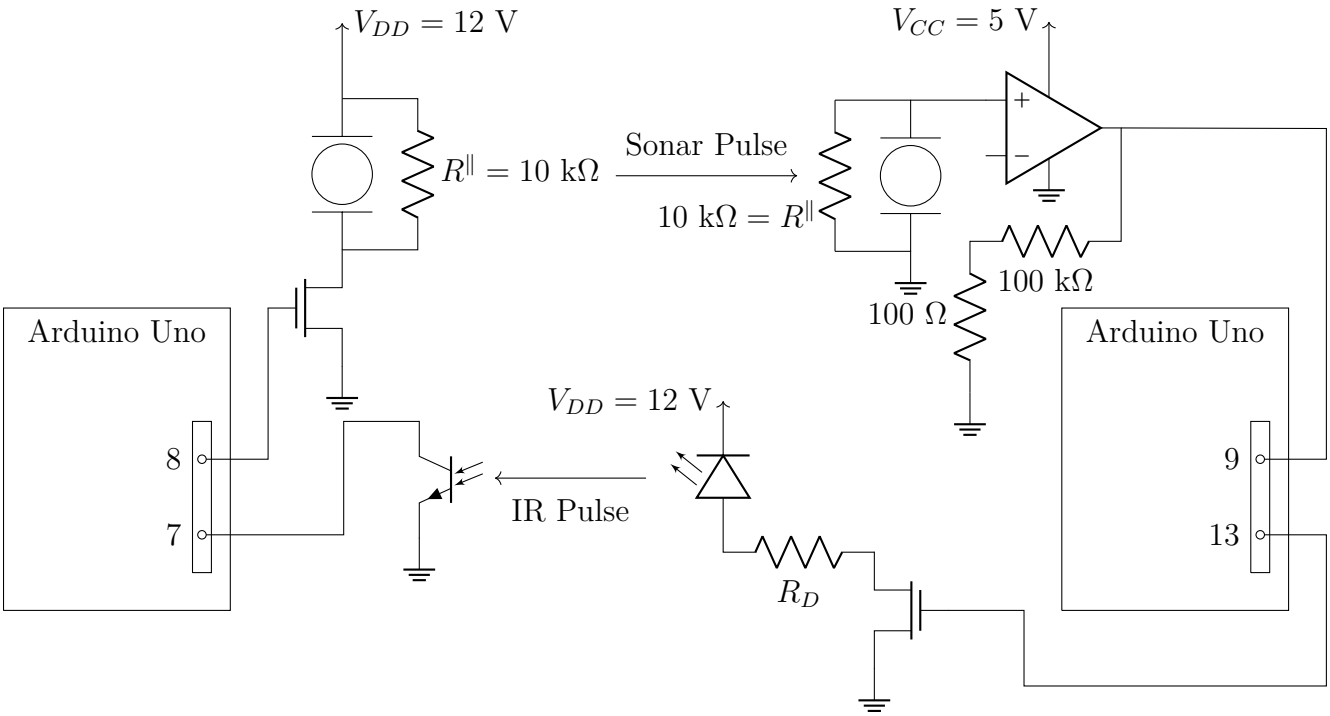

Figure 1.5: Initial Design for Ranging Schematic

There are a couple of key observations to note about this circuit. Both emitters are powered by 12 V, with an on-off toggle implemented with an Arduino digital output tied to an NMOS transistor gate. When the output is high, current flows from V_{DD} to ground activating the sonar/IR emitter. When the output is low, no current flows and the signal

ceases. On the receiver side, the IR receiver simply is fed directly into a digital input pin. When the IR signal hits the phototransistor, a voltage is induced across the collector and emitter (this time the transistor terminal, not the component responsible for emitting the signal). The sonar signal, conversely, needs amplification. The opamp configuration shown in 1.5 is designed to step up the sonar receiver voltage to levels readable by the Arduino digital input. This design served as the starting point for the investigation of sensor-integrated transport system.

Chapter 2

Methods

This chapter addresses the design and implementation of the system. A detailed analysis of design decisions made in the process of reaching this system can be found in Chapter 4. By the end of this chapter it should be clear how the robot's software, computer hardware, and auxiliary circuits are structured to implement sonar-infrared beacon sensing. The beacon design, albeit simpler, will also be covered in this chapter. Given that my team's focus was primarily on the drive system that will be the principal topic in this chapter, though the complete design picture will be clear. The design of this system should stand in stark contrast to that described in section 1.2, in both capability and efficiency of design.

2.1 Primary Onboard Compute Unit

This section details the redesign of the control system, from the layered approach shown in figure 1.3 to a consolidated approach built using a Raspberry Pi. In contrast to an Arduino microcontroller, the Raspberry Pi is a full-fledged personal computer, efficiently packed into a similarly sized board. The Raspberry Pi runs a fully capable OS (operating system) based on a modified version of the linux kernel, and is fitted with HDMI ports for display, Bluetooth/WiFi connectivity, USB ports for typical I/O (i.e. keyboard/mouse),

and an array of GPIO (general purpose I/O) digital output pins, some fitted with hardware generated PWM signal capabilities. This is all in addition to the memory virtualization, multiprocessing, and file system management capabilities that come with even the most lightweight OSes. The Raspberry pi is also easily mounted on the robot, in contrast to the Windows desktop used as the primary compute unit in the legacy design.

The software, implemented in Python3, is divided up into a hierarchical structure of packages and modules with an extremely simple dependency graph. Each relevant module or module group receives its own subsection in this section, which culminates in a description of the complete software system.

2.1.1 Driver Core

The driver core module implements layer 2 of the legacy system. It encapsulates the hardware system used to drive the robot as well as other parameters, including max forward/rotational speed and the frequency of the PWM signals. The module contains a single class `Driver` with a single method `drive`. That method takes parameters `turn_speed: float` and `forward_speed: float`. These parameters have no physical meaning, but merely control the forward translation rate and It has no return value, as it represents the bottom layer of the software system, but rather simply sets the duty cycles and H-bridge values for each wheel using a designated GPIO output pin. The driver is also responsible for initializing its GPIO pins properly, that is setting the two pins used to set the H-bridge to digital outputs, and initializing GPIO PWM instances for each PWM pin (as well as setting them to digital outputs) to support hardware PWM generation. Notably this occurs on a continuous time basis: the driver core module does not implement any timing aspects, it simply modifies the robot speed upon invocation of the drive method. A higher level system that invokes the driver core driver is therefore responsible for updating the forward and rotate speeds at relatively frequently, as otherwise the robot will likely oversteer and/or collide with an obstacle.

2.1.2 Higher Level Drivers

In theory, a driver core driver object could be instantiated explicitly from the python interpreter i.e. `d = driver_core.Driver()` and controlled directly via a series of calls to `d.drive(rot_speed, fwd_speed)`, but this does not constitute a usable system. The simplest interface in the codebase to the driver core is the keyboard driver module, which dictates forward and rotate speed via keystroke detection of a connected keyboard. A Bluetooth keyboard was connected to maximize mobility, and our wheelchair chassis achieved the capability of a simple remote control vehicle. The keyboard driver module has a single class Driver again with a single method drive. The keyboard driver instantiates a core driver and simply invokes `coredriver.drive()` with the appropriate parameters given the keyboard state. The other interface to the driver core is an autopilot: IR driver. the IR driver module integrates with the seven sensor IR module to orient the robot in the direction of the beacon, and drives towards it at a fixed speed

Higher level drivers are also responsible for integrating with sensor systems. This section covers only the software component of the sensor systems as well as their integration with the Raspberry pi, but a comprehensive description of sensory methods is included in section 2.2. The keyboard driver and IR driver each optionally has the ability to integrate with two sensor packages: obstacle avoidance (based on ultrasonic sensing) and camera localization. If integrated with obstacle avoidance, an additional layer is added between keystroke information or IR heading and the rotate/forward speeds passed into driver core. The obstacle avoidance module retrieves a series of four distances, each from an ultrasonic sensor at a different vantage point on the robot. The obstacle avoidance module then slows the forward/rotate speeds to avoid collisions if objects are detected in the direction of the robot's motion. The camera localizer logs position data based on tracking of key features on the ceiling of the room, but doesn't affect the drive system otherwise.

2.1.3 Sensing Modules

This subsection concerns the two primary analog sensing modules: obstacle avoidance and seven sensor IR. The obstacle avoidance module contains a single class: `Obstacle Avoidance` that is responsible for polling a COM (USB) port for four comma separated values corresponding to distance from the front right, front left, front, and rear bumpers. Those four distances are used to slow the right rotate, left rotate, forward, and backward speeds respectively if the robot is moving in that direction while the distance goes below a minimum. The rolling minimum over a sliding window of 25 measurements was used as the ground truth distance to avoid false "all clear" signals resulting in collision. The control mechanism applied scales speed like the following:

$$v = \begin{cases} -v_{max}/10 & d < d_{min} \\ v_{max}\left(\frac{d-d_{min}}{d_{max}-d_{min}}\right)^3 & d_{min} < d < d_{max} \\ v_{max} & d \geq d_{max} \end{cases} \tag{2.1}$$

Where v_{max} is the maximum speed, d_{min} is the shield distance, and d_{max} is the distance beyond which obstacle avoidance is no longer applied. Although this is clearly applicable to forward/backward speeds, it is also used for rotational avoidance as well using the side-bumper distances and rotational velocities. -v_{max}/10 is used to stop the robot by reversing its gears, simply setting speed to 0 would let the robot roll into the obstacle!

The second sensing module of interest is the seven sensors module. It contains a single class SevenSensorIR that encapsulates the GPIO-level details of IR readings. The SevenSensorIR class has two methods: get IR and get IR PWM. getIR simply returns an array of boolean values corresponding to whether each of the 7 IR sensors (arranged in a crown shape) has detected a signal. This is used to detect the heading of the beacon for orientation purposes as mentioned in the previous section. Get IR PWM was not used in this integration but would allow for the encoding of beacon identity in the IR signal. By polling the

IR sensors over a period and detecting the duty cycle, the robot could differentiate between at least two beacons. The infrared package (which contains the seven sensors module) also contains utility modules for visualizing sensor data and testing the system.

2.1.4 Camera Localizer

Though not integrated into the robot's beacon sensing/auto-drive capabilities, the Raspberry Pi camera was included as a localization aid. By pointing the camera at the ceiling, the camera tracked the motion of key features and used that transformation to track the movements of the robot. The camera tracked two types of key points: features detected by ORB (Oriented BRIEF, or Oriented Binary Robust Independent Elementary Features), and the intersection of lines detected by Hough transforms. ORB is a feature detection method implemented in the Python Opencv library for free use that takes an image as input and generates a set of key points. Using the builtin descriptor matcher, a mapping from the key points in a prior image to those of the current image was created (Karami et al. 2017). The Hough transform (or particularly the probabilistic Hough transform) is an algorithm with various applications that is used for line detection. In the context of images, the Hough transform generates a probability matrix of whether a pair of points lie on the same line in the image, and then return the best few lines as determined by a set of hyperparameters. The pairwise intersection intersection points between (roughly) perpendicular line pairs were then computed and greedily clustered into sets with a maximum radius of 15 pixels. The means of those clusters were then included in the feature set, and matched with the previous image by minimizing distance.

Once two equal cardinality point sets were obtained, one in the prior image and the other in the current image, the transformation between the point sets was fitted to a translation and a rotation using singular value decomposition (Arun et al. 1987). Finally after the translation vector and rotation matrix were obtained, the changes in position were applied in the current frame, and then transformed back into the original coordinate system to

accumulate position data. In practice the rotation angle derived from this method performed poorly, so the alternative method of relying on the grid-like structure of the ceiling was used for angle tracking. Everything else, though, remained the same.

2.1.5 Complete Software System

Now that each component of the software system has been outlined in detail, it is fitting to put the pieces of the puzzle together as seen below:

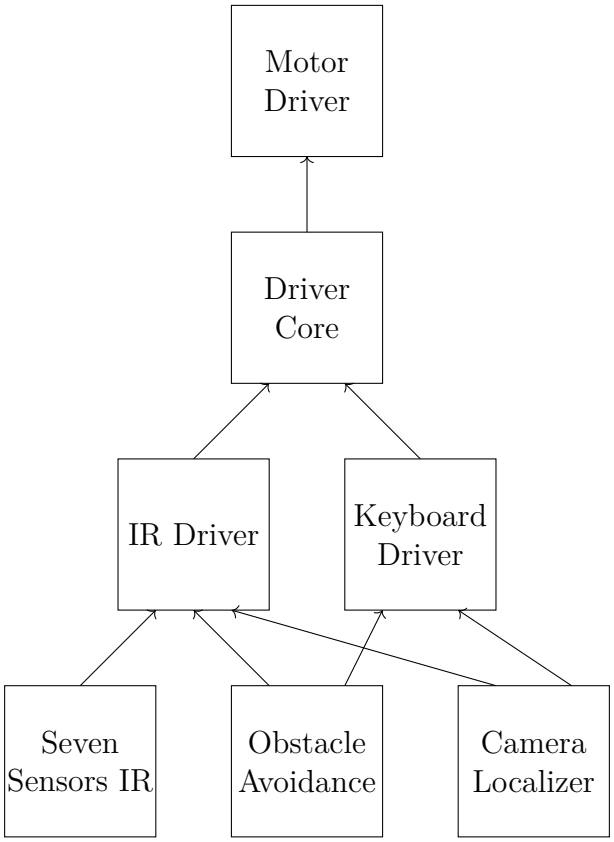

Figure 2.1: Dataflow Diagram of Integrated Software System

Note that the IR Driver and Keyboard Driver are the modules that are directly used, which can be deceptive given their position in the center of the dataflow graph. An intuition can be salved from the fact that these objects intake information from the sensors and output it into the core driver, which in turn drives the robot. This is hopefully a convincing argument for the usage of IR Driver and Keyboard Driver as top level control units.

2.2 Sensor Systems

Though not my area of focus, I will attempt to survey the external sensor systems used to aid the drive system for the sake of completeness. Note that this excludes the camera system, as that is more or less built into the Raspberry Pi and required zero external circuitry or integration. The systems surveyed here are the off-board ultrasonic system for obstacle avoidance, the infrared sensing circuit behind the seven sensor IR module, and the beacon design. Also mentioned is a rough overview of a custom sonar system that was still in development at the end of the semester, so was never integrated into the robot.

2.2.1 Ultrasonic Obstacle Avoidance System

In order to implement obstacle avoidance, four ultrasonic sensors were deployed to prevent running into (or backing into) an obstacle directly head on, as well as rotating into an obstacle directly to the side of the robot at the time of rotation. The design for a single ultrasonic circuit is shown below:

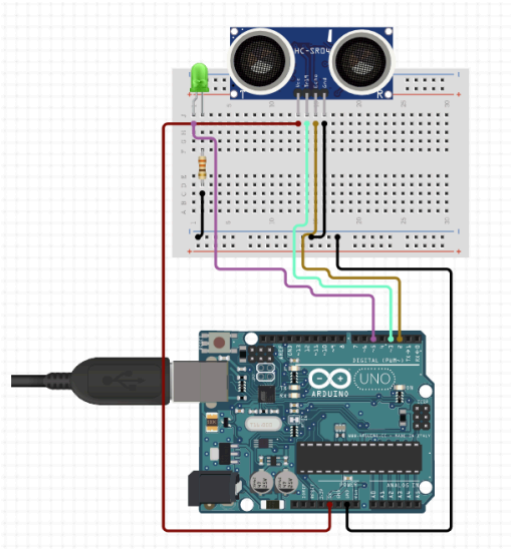

Figure 2.2: Ultrasonic Circuit Design

The resistance value used was 1 kΩ. Rather than using four separate microcontrollers,

we used a single Arduino to power all four ultrasonic circuits and synchronized their pulses to reduce cross interference between signals. The Arduino then performed ToF analysis, and sent the list of ranges as a comma separated Utf-8 encoded string out through the COM port. As discussed earlier in section 2.1.3, the Raspberry Pi was configured to poll the COM port waiting this information. The ultrasonic circuit was set up to record 100 measurements per second, sufficient frequency for fine grained robot control but a sufficiently low frequency for accurate ToF readings. The obstacle avoidance mechanism given recorded distances is also discussed in the previous section.

2.2.2 Seven Sensor IR Apparatus

The eyes of the robot, the seven sensor IR system was a crucial component of the project. The IR group designed a 3D-printed set of IR sensors lain out at angles ranging from roughly -75 to 75 degrees off of the robot's field of view. The IR circuit was based on the IR receiver design in the introduction and shown in figure 1.5. The design utilized unamplified phototransistor signals as inputs to digital GPIO pins. The usage of these digital signals is discussed in the previous section. The group faced numerous challenges in terms of current consumption when scaling up, and navigated them expertly to produce this result.

2.2.3 R&D: Custom Sonar System

The sonar group attempted to create a custom sonar circuit that would exceed the range and beam-width of the existing ultrasonic components. The ultrasonic sensor used in the obstacle avoidance circuit is mounted on a complex board, which the group attempted to improve upon. Using discrete semiconductor components including comparators, timers, and opamps, the group struggled with varying power tolerances as they attempted to operate the circuit at high voltage and current to push the limit of range. Unfortunately this system never made it over the finish line in time to integrate with the robot, though it remains an area of interest for further research.

2.2.4 Infrared Beacon

As the sonar system was never integrated into the beacon as in the proof of concept design, the beacon was designed simply to provide an IR emitter for orientation. Distance sensing would then have to be done with the on-robot ultrasonic sensors, which would in turn double as beacon range sensors under the assumption of an open landscape. The infrared beacon was designed as a wall-outlet compatible high power infrared emitter mounted on a stepper and backed by an angled surface for signal reflection (similar in shape to a modern satellite dish). An image is shown below for reference: With the IR sensing and emitting systems completed

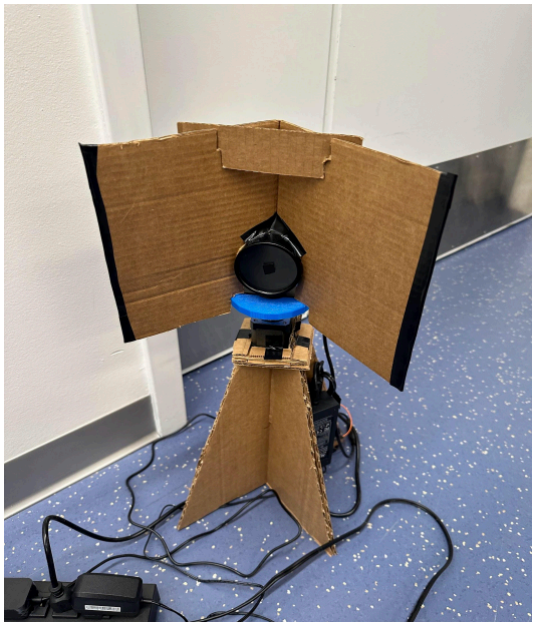

Figure 2.3: Photograph of Infrared Beacon

and integrated into the drive software, the robot was poised to perform autonomous indoor navigation toward the beacon.

Chapter 3

Results

This chapter surveys the results of implementing the aforementioned design. The chapter evaluates the robot's ability to perform each of the following tasks:

(a) Mitigate collision risk under manual drive

(b) Navigate toward and stop in front of an IR beacon in an open environment

(c) Localize using computer vision

Due to the challenges with sonar sensing explained in the previous chapter, the robot was unable to perform localization based on the singaround-based design with multiple beacons. However, the robot achieved a high degree of success with both collision avoidance under manual drive as well as beacon-seeking autonomous navigation. The camera localizer produced extremely mixed results, but is a promising approach given more time to iterate.

3.1 Manual Drive Collision Avoidance

When the keyboard driver object was instantiated with obstacle avoidance enabled, the sonar sensors were effective in avoiding head on, rear on, or simple rotation collisions. Simple rotation collisions here are defined as stationary rotation into a large object that remains in the field of view of the side-facing sensor throughout the rotation. The difficulty with non-

simple rotations as well as collisions with objects not directly in front of the robot's center comes from coverage. The robot chassis is relatively large, so especially at close distances the ultrasonic system is riddled with blindspots. The results are promising, though, as it seems increasing the number of sensors (which is cheap both economically and power-wise) would fix the issue.

3.2 Beacon-Seeking Autonomous Navigation

As long as the robot's starting orientation was within 180 degrees of the beacon's center of oscillation, the robot was able to detect and orient toward the beacon, before driving up to it and stopping within some distance. There were a few issues that the group successfully resolved in the process of fine tuning this process. First was that the rotating beacon would cause the robot to rotational oscillation even at the minimum drive distance. This is because the obstacle avoidance feedback mechanism only stops forward motion unless there is an impedance to the side, otherwise it'd be impossible to escape a wall! In this case we applied a tweak given that the distance to a forward obstacle was presumed to be the distance to the beacon: also scale down rotate speed as the robot approaches the beacon. This had the effect of winding down the "wiggle" as the robot neared the destination.

The second issue was that the original placement of the front-facing ultrasonic hit the beacon at a thin point. This caused occasional ultrasonic misses even as the robot neared the beacon, ending up with a collision! Re-placing the sensor at a higher point even with the large reflective surface alleviated the problem. Overall the robot's performance on this task can be characterized as a success, though future work is definitely needed to generalize this demonstration to more applicable autonomous navigation.

3.3 Camera Localization

The camera based localizer demonstrated extremely mixed results. On rich regions of the ceiling it performed extremely well, with very few hiccups between snapshots. The issue is that the error here builds, as the position is taken as the composition of inter-photo transformations. Therefore a bad reading can skew the position data for the rest of the run. One of the big performance upgrades was turning off the ceiling lights so that the camera could better focus on the regions that were otherwise totally darkened in contrast to the dominant light source. Below is a set of images demonstrating the image processing pipeline: From left to right we see the raw image, a black and white image derived from

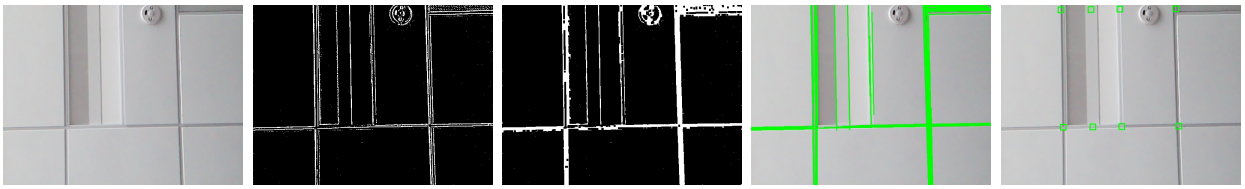

Figure 3.1: Image Processing Pipeline

dynamic thresholding, a clearer black and white image derived from morphological kernel processing, the raw image with Hough lines superimposed (derived from morphologically processed image), and finally the clustered intersection points. This point set is sufficiently rich for analysis, and therefore performed well in path tracing. In other instances, though, there were one or no total points in the set, which failed to provide sufficient information to determine location change. Augmenting the intersection point set with ORB features provided mixed results: for one the dataset size improved on average, but the quality of the mapping was drastically worse for the ORB keypoints than it was for the Hough intersections. Despite many failures, there was promising data from one of the most successful tests of the whole process shown below. The left figure shows timeseries x and y data from the path, while the right plot traces out the coordinates in the Cartesian plane. It is apparent that the camera system does a good job of grasping the general shape of the path and accumulates

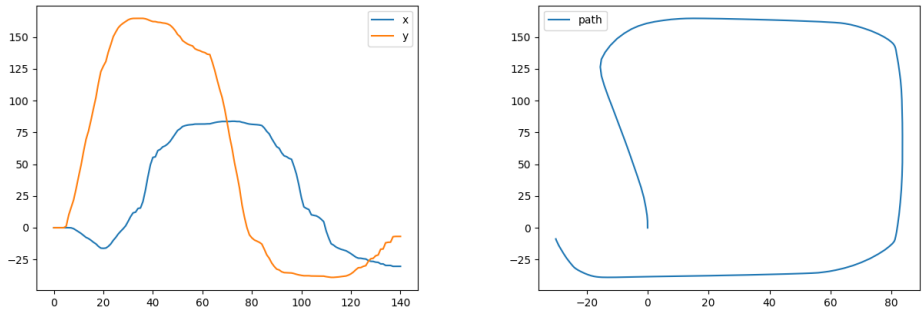

Figure 3.2: Localization Data From Roughly Square Path

minimal error. The path also may not have exactly closed in reality, so it's hard to tell the exact magnitude of the true error. This was the best test recorded, though, so it's clear there is room for improvement. The camera is promising though as a means of localization, as there are a whole host of techniques that can be used in various camera positions and image processing methods. Overall the camera localization experiment was unreliable, but showed signs of promise as a subject of future work.

Chapter 4

Design Decisions

This chapter focuses on the various design decisions and engineering tradeoffs encountered in the process of reaching the design described in Chapter 2. Topics include the headless operation of the Raspberry pi, the timing interval between snapshots of the camera localizer, GPIO/Arduino pin allocation between sensory applications, the overall structure of the codebase, and the particular obstacle avoidance response implementation. Excluded are the various design considerations of the sensor teams, as I did not have much insight into those processes.

4.1 Headless Raspberry Pi Design

There were two main drawbacks to the MATLAB-desktop-workstation configuration of the legacy system. First off, the MATLAB script controlled the robot in discrete timesteps which was untenable for a smooth autopilot experience. This was changed purely in software: now the top level control unit updates robot state rather than setting, delaying, and resetting. This enables nearly continuous-time drive by updating the robot state in a loop. The focus of the section is therefore the elimination of wired connections from the Raspberry Pi to a controlling workstation. When the desktop computer had to be plugged into the wall,

this was not an issue in and of itself. Now that the Raspberry Pi is mounted directly on the robot and powered by the onboard inverter, the wires from the pi to the controlling keyboard, mouse, and monitor are the limiting factor of robot mobility. These connections were then promptly removed by taking advantage of the Raspberry Pi's network capability.

4.1.1 Networked Solutions for Control

In order to manage the running scripts/processes on the Raspberry Pi remotely we take advantage of the SSH (secure shell) protocol, which grants remote access to a bash terminal over an internet connection. This allows for full management of the Raspberry Pi through the CLI (command line interface). When debugging it became apparent that the Raspberry Pi was not completely properly configured for networking. However, when both the client laptop and the Pi were connected to Yale Guest network (Yale Secure was too tricky) SSH worked seamlessly. The one functionality that SSH leaves out is keyboard control. Keystrokes from an SSH client appear in the bash terminal but don't register with the SSH server's (the Pi's) OS. This means that the Python keyboard library fails to detect keystrokes from an SSH client– an alternative pathway is needed.

In order to control the robot over the network a Bluetooth keyboard was introduced. The Bluetooth keyboard connects directly to the Pi rather than via SSH, so keystrokes on that device can directly control the robot. Also, it is simple to manage the Bluetooth service over the CLI, so the keyboard can be connected without ever needing a wired monitor/mouse.

4.1.2 Networked Control System Design

This subsection concerns an analysis of the solution at the system level, beginning with a diagram: It is a noteworthy fact that the "start/stop" functionality is communicated over a completely different pathway (SSH) from the control once underway. This gives rise to interesting scenarios during testing when the Bluetooth goes out but the internet stays up, or vice versa as the system isn't up or down, but rather in an intermediate unstable state.

Figure 4.1: Network Diagram of Control System

4.2 Optimizing Camera Snapshot Interval

This section concerns the optimization of camera snapshot interval. After software optimization, computer vision processing time was no longer a bottleneck– images could be sampled at a nearly arbitrary rate and passed through the pipeline for positioning. There are two main constraints on processing rate: mapping complexity and signal noise ratio. If images are sampled too far apart, then mapping points in clusters from the first image to points in clusters of the second becomes exceedingly difficult. The minimal distance map decays in accuracy, and more advanced methods also begin to fail. If the sampling rate is too high, conversely, then the signal to noise ratio falls off: the points may have moved 2 pixels in reality but have a 10 pixel noise margin in the image. After an iterative process, it was determined that sampling images at 10Hz gave optimal results.

4.3 Pin Allocation/Distributing Compute

One of the notable pieces of this design is that there are two compute units: a primary unit (the Raspberry Pi) and an offboard microcontroller (the Arduino). This design choice was made for a couple of reasons. The obvious one is that the Raspberry Pi has insufficient digital pins to handle all of the sensors, so some sort of split is necessary. In theory it is possible to extend the digital pin structure using shift registers, but this was deemed an unnecessary complexity. Consolidating all of the ultrasonic sensors onto one Arduino seemed to be a logical design: the Arduino has ample compute power to handle the simple calculation of distance from ToF, and is useful to synchronize the pulses of the four ultrasonic sensors.

The Arduino can then easily communicate these results back over the COM port. Overall moving the ToF and digital pin overhead off of the Raspberry Pi is a very fruitful design decision that improves the overall quality of the system.

4.4 Software Module Structure

A massive portion of the design process of the drive system came down to the structure of the codebase. The software was carefully implemented according to an object oriented paradigm as well as a directional dataflow model from sensors to drivers to the core to the robot. Much of this is discussed in previous sections so further elaboration would likely be redundant, but this was an extremely important piece of the design process.

4.5 Obstacle Avoidance Strategy

Early on in the process we had a conversation with Professor Kuc about obstacle avoidance. Though many strategies use feedback to exert what amounts to a repulsive force between the vehicle and the obstacle, these strategies take control away from the user and add uncertainty to path planning processes. With this in mind, we elected to use a slowdown based approach to obstacle avoidance. This approach gives the user the ability to respond to their nearing an obstacle without being swerved out of the way, slowly reducing their speed to a stop if they do not respond.

Chapter 5

ABET Outcomes

This chapter concerns the alignment of this project with the ABET outcomes. ABET defines a set of core objectives for students, which this project was designed to fulfill. Each section will address one of these "outcomes", and paint a narrative of demonstrated competency from this project.

5.1 Engineering Problem Solving

> "An ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics."

In this project, students were tasked with creating an autonomous vehicle that navigated indoor environments using sensory information. This could be the epitome of a complex engineering problem, one that billions of dollars and millions of man hours are being thrown at as we speak. The solution that the group designed requires expertise in the domain of electromagnetism, acoustics, and obviously electronics engineering. In the end the robot was able to successfully seek an infrared beacon, mitigate collision risk, and there are clear pathways toward accurate beacon-based and camera-based localization strategies.

5.2 Health, Environmental, Economic Considerations

> "An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors."

As illustrated in Section 4, there are various factors that go into every decision made in the process of engineering a system. The design decision that best demonstrates competency in this area in my opinion is the obstacle avoidance response system. The group responded to the issue of injuries to grandchildren and pets while keeping control in the user's hands. This was also accomplished while keeping economic costs low, utilizing a single low-cost microcontroller and four relatively cheap ultrasonic sensor components.

5.3 Communication With Range of Audiences

> "An ability to communicate effectively with a range of audiences"

Over the course of this project the group was required to communicate effectively both among ourselves as well as to the outside world. This report is designed to reach a variety of readers, ranging from experts in the field of sensing and robotics to young students interested in learning more about STEM. Internally, members of the group had a vast range of backgrounds– I myself have a stronger programming/computer science background while others excel in circuit design, soldering, and analog signal processing. Communication between groups and backgrounds was integral to the success of the project.

5.4 Ethical & Professional Responsibilities

> "An ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts."

The team demonstrated a high degree of professionalism and ethical considerations throughout the project process. Students constantly made time to meet and hack through problems together despite busy schedules and various commitments. In the design ethics were considered in the drive speeds and obstacle avoidance systems which were optimized for quality of life and safety. Students also demonstrated a high degree of responsibility and personal accountability in the lab, making sure all equipment was returned to its original place and batteries plugged in before leaving the lab.

5.5 Teamwork & Collaboration

> "An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives."

I can attest to the importance of effective teamwork. Our team was made up of individuals with different skill sets and backgrounds, which allowed us to approach the project from a variety of perspectives. We established clear goals and planned tasks accordingly, delegating responsibilities based on each team member's strengths. Our inclusive and collaborative environment allowed for open communication and constructive feedback, which led to a more cohesive and polished final product. By working together towards a common objective, we were able to meet our project objectives and create a successful robot that utilized sonar and infrared technologies. Overall, this project demonstrated the power of teamwork and the importance of creating a supportive and collaborative environment to achieve shared goals.

5.6 Experimentation & Data Analysis

> "An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions"

Developing and conducting appropriate experimentation, analyzing and interpreting data,

and using engineering judgment to draw conclusions is a crucial principle in engineering. In this project the group was tasked with evaluating the viability of various sensor based approaches to indoor navigation. We developed experiments to evaluate the ranges and beam widths of various sensors, as well as test the path planning capabilities of the different systems. Our engineering judgment allowed us to weigh the benefits and drawbacks of the configuration and make informed decisions. This project highlighted the importance of developing a robust experimental design, collecting and analyzing data effectively, and using sound engineering judgment to draw conclusions. It demonstrated how these skills are essential for making informed decisions that can have a significant impact on a project's success.

5.7 Knowledge Acquisition & Learning

> "An ability to acquire and apply new knowledge as needed, using appropriate learning strategies."

In the constantly evolving field of engineering, the ability to acquire and apply new knowledge using appropriate learning strategies is critical. Going into this project, I had limited experience with designing systems compatible with Raspberry Pi/resource constrained systems. I was used to having arbitrarily many CPU and GPU cores available as needed, but the Pi was limited to a single CPU. Adapting to this resource constraint required fast learning and usage of online resources/research papers. Additionally, the analog aspects of this circuit were extremely new, and demanded similarly agile knowledge acquisition. This project demonstrated the importance of being adaptable and willing to learn new skills and technologies. By being proactive in acquiring and applying new knowledge, engineers can stay at the forefront of their field and deliver innovative solutions to complex problems.

Chapter 6

Conclusion

This project was a journey, culminating in a robot that had both autonomous and aided manual driving capabilities for navigating both open and obstacle ridden landscapes. It's definitely unfortunate that the camera based positioning system never came together in a reliable sense, but it represents an interesting opportunity for future work and showed a lot of promise. It was an awesome moment to finally get the robot working– seeing it inch toward the beacon while reorienting itself to the spinning IR emitter was a real joy. There were moments of frustration when it felt like the drive system was implemented but there were no working sensors to integrate, but everything came together in the end. As usual, the little things hurt too– connecting the Bluetooth keyboard and figuring out how to turn the lights off have taken hours off my life that I'll never get back. I'd also love to have finished the sonar shield with more complete coverage, either by increasing the number of sensors or augmenting the system with servo motors. With the localization system and ultrasonic shield complete autopilot is only a programming project away. The project was an fascinating exercise in augmenting an existing system with sensors and computers– I'm excited to see where the next cohort of students take it from here.

Bibliography

2020. 2020 sps performance standard - gps: The global positioning system. URL https://www.gps.gov/technical/ps/2020-SPS-performance-standard.pdf.

2023. Autonomous vehicle market (by application: Defense, transportation; by level of automation: Level 1, level 2, level 3, level 4, level 5; by propulsion: Semi-autonomous, fully autonomous; by vehicle: Passenger car, commercial vehicle) - global industry analysis, size, share, growth, trends, regional outlook, and forecast 2023 - 2032. URL https://www.precedenceresearch.com/autonomous-vehicle-market.

Arun, K.S., T.S. Huang and Steven Blostein. 1987. Least-squares fitting of two 3-d point sets. ieee t pattern anal. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-9. 698 – 700. doi:10.1109/TPAMI.1987.4767965.

Flueratoru, L., S. Wehrli, M. Magno, E. S. Lohan and D. Niculescu. 2021. High-accuracy ranging and localization with ultra-wideband communications for energy-constrained devices.

Flynn, A.M. 1988. Combining sonar and infrared sensors for mobile robot navigation. *The International Journal of Robotics Research* 7(6). 5–14. doi:10.1177/027836498800700602. URL https://doi.org/10.1177/027836498800700602.

GIS. 2023. Everything you need to know about gps l1, l2, and l5 frequencies. URL https://gisresources.com/.

Guo, Hua, Mengqi Li, Xuejing Zhang, Xiaotian Gao and Qian Liu. 2022. Uwb indoor positioning optimization algorithm based on genetic annealing and clustering analysis. *Frontiers in Neurorobotics* 16. doi:10.3389/fnbot.2022.715440. URL https://www.frontiersin.org/articles/10.3389/fnbot.2022.715440.

Harder, Paula, William K. Jones, Redouane Lguensat, Shahine Bouabid, James Fulton, Dánell Quesada-Chacón, Aris Marcolongo, Sofija Stefanovic, Yuhan Rao, Peter Manshausen and Duncan Watson-Parris. 2020. Nightvision: Generating nighttime satellite imagery from infra-red observations. *CoRR* abs/2011.07017. URL https://arxiv.org/abs/2011.07017.

Karami, Ebrahim, Siva Prasad and Mohamed S. Shehata. 2017. Image matching using sift, surf, BRIEF and ORB: performance comparison for distorted images. *CoRR* abs/1710.02726. URL http://arxiv.org/abs/1710.02726.

Phon-Amnuaisuk, Somnuk, Ken T. Murata, La-Or Kovavisaruch, Tiong-Hoo Lim, Praphan Pavarangkoon and Takamichi Mizuhara. 2022. Visual-based positioning and pose estima-

tion.