Hardware Synchronization System

Project Proposal

Sensory

Ritwik Sathe

Wyatt Hansen

Matt Boyett

Colton Mulkey

Michael Mengistu

Department of Computer Science

Texas A&M University

Date: 9/28/21

Table of Contents

Executive summary

The overall goal is to implement a system that triggers the collection and synchronization of camera, radar, and lidar sensor data. The sensor data collection is currently not triggered and synchronized by a single system, so the autonomous vehicle is not able to detect potential hazards ahead. The objectives are to achieve an error rate below a set amount and to create a system that eventually runs in real time rather than a short amount of time.

The overall project design will feature several hardware and software components that will work together to timestamp and synchronize sensor data. For the software component, we will develop a program for the Arduino that will take the PPS signal output from the GPS as a serialized input. The Arduino will then modify the PPS signal and send it to the camera, lidar, and radar sensors to ensure synchronization of all components. The components will then output their respective 10 Hz signals to a Ubuntu PC, which will be receiving a separate 10 Hz and 1Hz signal from the Arduino as well. When the PC has received all of these signals as inputs, it will then send off the data, timestamps, and error analysis to memory to be stored. Following this, the data captured will either be manually evaluated or automatically evaluated to determine the error rate utilizing a ROS-based program we will develop.

For the Hardware component, the GPS, Arduino, sensors, and PC must have all their connections properly wired. This will require proprietary connectors to be used in our system to allow for proper interfacing between both the input and output ports of the sensors. Along with port connectors, the GPS output will need to be connected to the Arduino input port as well. A casing will also need to be built for the Arduino to keep the hardware safe from any damage.

The results of this project will yield an efficient way to synchronize different sensors together. This allows for the synchronized data to be effectively used in a car or robot system which makes every action it takes more accurate and safe. This solution will be able to be reused for future projects with different systems saving both money and time on future robot or car projects. Overall, this system design is necessary for precise and accurate information from multiple sensors and allows the rest of the system to operate significantly better.

1 Introduction

The general scope is to develop a system that uses the Pulse-Per-Second (PPS) signal from a Global Positioning System (GPS) to synchronously trigger sensors (camera, radar, and lidar sensors) in an autonomous vehicle. A PPS signal is what the name suggests: a very short signal with a sharp rising or falling edge that repeats every second; it is then obvious that the frequency is 1 Hertz (1/second). GPS-based time has long been trusted to provide extremely accurate time for device synchronization. Such accuracy is crucial if the devices are to work together to navigate the vehicle. Finally, Robotic Operating System (ROS) development software will synthesize time-stamped data collected from the sensors to minimize inconsistencies between the data. This will ensure proper decision-making by the autopilot system. Several teams are developing the sensors' capabilities, and the success of their work depends on how well the sensors are integrated and synchronized with minimal error. Thus, this division serves as a crucial intermediary between the external signal and the proper data collection.

1.1 Needs statement

The various sensor data collection is currently not triggered and synchronized by a single system, so the autonomous vehicle is not able to detect potential hazards ahead.

1.2 Goal and objectives

The goal is to implement a system that triggers the collection and synchronization of camera, radar, and lidar sensor data. The precision of the synchronized data will help the autonomous vehicle accurately detect the presence of hazards and traffic controls in order to keep its passengers safe. One stretch goal is to implement alerts whenever a sensor stops working or starts malfunctioning. If that goal is possible, passengers can be notified of the issues and get them so that they do not endanger themselves in a car with bad sensors. The progress will be measured against the following metrics: physical development, running time, mean and standard deviation of synchronization delay, and synchronization error.

1.3 Design constraints and feasibility

Within the system, all hardware is connected through native serial connections, so data transfer will be instantaneous. The only wireless data link is between the GPS and the Arduino framework. The transfer pulses occur on the rising edge of each clock cycle. While the camera and radar sensors can accept 10 Hz pulses, the Velodyne lidar sensors can accept only 1 Hz pulses, so the lidar sensors will have less data to collect. All sensors will still transmit data at 10 Hz to the computer. The main challenge is to program the computer to match the time-stamped data amongst all devices and overcome the data collection time window discrepancy. While PPS time signal outputs may be extended to synchronize at smaller time scales (see 3.4), the ability to achieve this depends on the circuit's susceptibility to clock skews, which is a phenomenon in synchronous digital circuit systems where the same sourced clock signal arrives at different components at different times. The skew is also referred to as the instantaneous difference between the readings of any two clocks. Because of the potential synchronization issues, the vehicle must have a maximum distance between itself and the obstacles to begin data collection and simultaneous data validation while traveling at a certain speed. Basic data validation will be necessary just to demonstrate that the hardware is synchronizing properly, while the detailed validations will be performed by the other teams that are developing the sensors' capabilities. Finally, the circuit must have minimal power consumption.

2 Literature and technical survey

The prior research and development of systems that trigger the collection and synchronization of sensor data is immense. These systems have been needed in a vast number of industries such as Unmanned Aerial Vehicles, Hobbyist projects, and high-speed camera photography. Below are some examples of these systems.

2.1 "Using PPS to Synchronize with External GPS"

2.1.1 This paper details a current product, FLIR's Ladybug5+ Camera, and its ability to directly accept PPS signals and NMEA data streams over the GPIO pins. By accepting the PPS signals, the internal precision time is continuously synchronized with GPS time and the timestamps of each image frame are within the microseconds of GPS time. The Ladybug API that FLIR developed has two functions that a synchronization system would use to set and get GPS time sync. [6]

2.2 "Time Synchronization for Wireless Sensors Using Low-Cost GPS Module and Arduino"

2.2.1 "The paper presents a new time synchronization method working independently on each node without exchanging time-sync packets among nodes." The authors show that through this method it is faster and more time-efficient. This can be done without building the wireless sensor network with each device. The system goes through a process of time stamping data, and re-samples this data to get time-synchronized data. They found that the time relative errors were found to be within +-400ns, while the time-stamp standard deviation was 40.8ns. [7]

2.3 "Open-Source LiDAR Time Synchronization System by Mimicking GPS-clock"

2.3.1 This paper evaluates the time synchronization of multiple sensors when building a sensor network. In the research paper they use open-source LiDAR to demonstrate how to synchronize multiple other sensors such as cameras, radars, and other types of LiDARS. They also mimic a GPS clock interface to provide a 1 µs synchronization precision for their time synchronization system. This research can be used to help design our project since we need to be able to trigger a Pulse-Per-Second (PPS) signal from a GPS to synchronize multiple cameras, lidar sensors, and radar sensors. [5]

2.4 "Generation of GPS Based Time Signal Outputs for Time Synchronization Application"

2.4.1 This research paper details an experimentation of time signal output from binary frames received from the GPS. It compares PPS output and serial communication output. The hardware was configured to receive both PPS and Serial interrupts and generate NGTS/T and NMEA-format packets according to the signal received. The research concluded that synchronization intervals can be stretched out or shrunk between 1 second and 1 minute. Particularly regarding PPS, the same design can be extended to achieve time precision in milliseconds using NTP and other algorithms. Regarding the proposed project, the research gave a better idea about the PPS time packet output that the system is expected to handle when synchronizing. [8]

2.5 "Brief industry paper: The matter of time — a general and efficient system for precise sensor synchronization in robotic computing"

2.5.1 "The Matter of Time" outlines synchronization systems in robots and the challenges and methods behind them. The article mentions an Intersection over Union (IoU) method which can be "used to determine the threshold of tolerable synchronization error." This would be one useful way to determine how effective our synchronization is. To achieve synchronization, the paper outlines the requirements which are, triggering sensors simultaneously and giving an accurate timestamp to each sensor sample. The difficulties with this being the differences between the sensors triggering mechanisms and time stamping mechanisms. The paper then proposes their solution which is to design specific hardware to trigger the sensors at the same time. Then Systems-on-chip is used to get an accurate timestamp on the data as soon as it exits the sensor. The paper explains that making

corrections to data through the robots OS will introduce more variance so their design focuses on using hardware. Overall, this paper describes a problem close to ours and offers up a good solution that could be implemented into our project. [9]

3 Proposed work

3.1 Evaluation of alternative solutions

- Arduino vs Raspberry Pi

Arduino	Raspberry Pi
● Microcontroller board with just CPU, RAM, ROM, and IO connectivity ● Only need a firmware instructing the hardware what to do ● 16 MHz clock speed ● Traditionally used for for interfacing Sensors and controlling LEDs and Motors ● Digital IO (for digital Input and Output) and Analog IN (analog input) pins ● Micro-USB connection but no power adapter needed (just a USB port) ● Power cuts simply cause a restart with no damage to hardware or software ● Uses Arduino IDE ● Models range from $4-$23	● Microprocessor-based mini computer with a CPU, RAM, storage, graphics driver, on board connectors, and an Operating System ● Need to work with Raspberry Pi OS ● 1.2 GHz clock speed ● Traditionally used for developing software applications w/ Python ● 40-pin GPIO for connecting LEDs, Buttons, Sensors, and Motors ● Micro-USB connection but power adapter needed ● Power cuts can damage hardware and software ● Can use any IDE ● Models range from $35-$75

While Raspberry Pi's are without a doubt more versatile and powerful, using a lightweight, low maintenance hardware such as microcontroller seems to be the smarter move since it can provide all the functionality needed with high fault tolerance and little need for maintenance.

- Data validation alternatives (manual vs automatic and Pendulum vs other dynamic systems that can be modeled)

  There are two main ways to do data validation either manually or automatically. Doing things manually comes with many drawbacks like the limited amount of data could be analyzed. It would also be subject to human error but it would not take much setup time to start validating data.

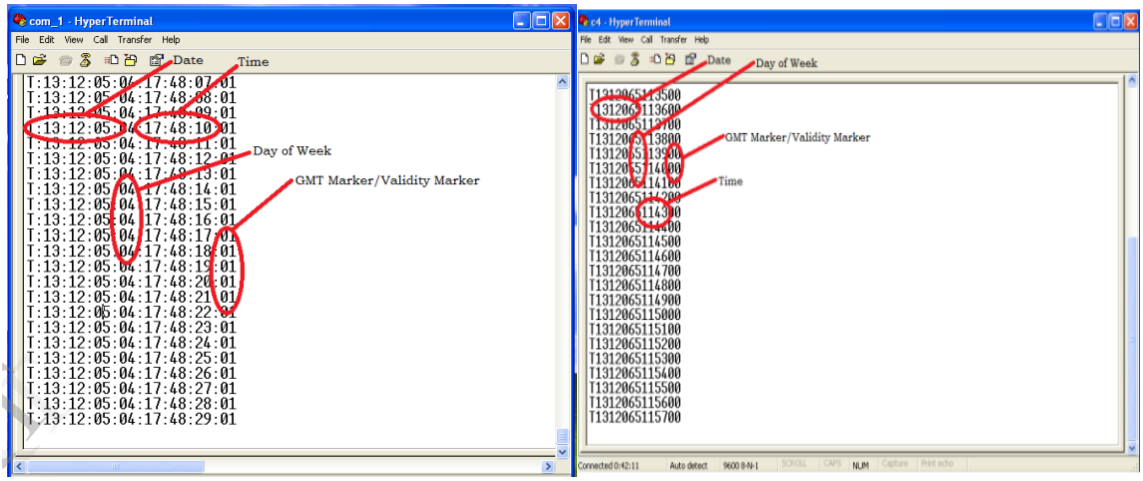
  On the other side of the coin, there is automation. Automating this validation process would allow us to analyze a large amount of data and be very precise. This comes at the cost of a large setup time to code an automatic solution. The main way to do this would be to grab an object which moves constantly and then measure the distance between where the object is at the same timestamp between different sensors. A great object for this would be the

pendulum, which has consistent movement despite slowing down over time. Overall, automating the data validation process would be much better than manual but it would take a longer time to set up and implement.

- Timestamp Validation

  Since the external signal is a GPS PPS signal, the timestamps may be output in one of two formats: National Grid Technical Standard (NGTS) frame or T frame. The following pictures comparing the two kinds of outputs are from the research paper for which the timestamp technical survey was based on:

T format [8] NGTS format [8]

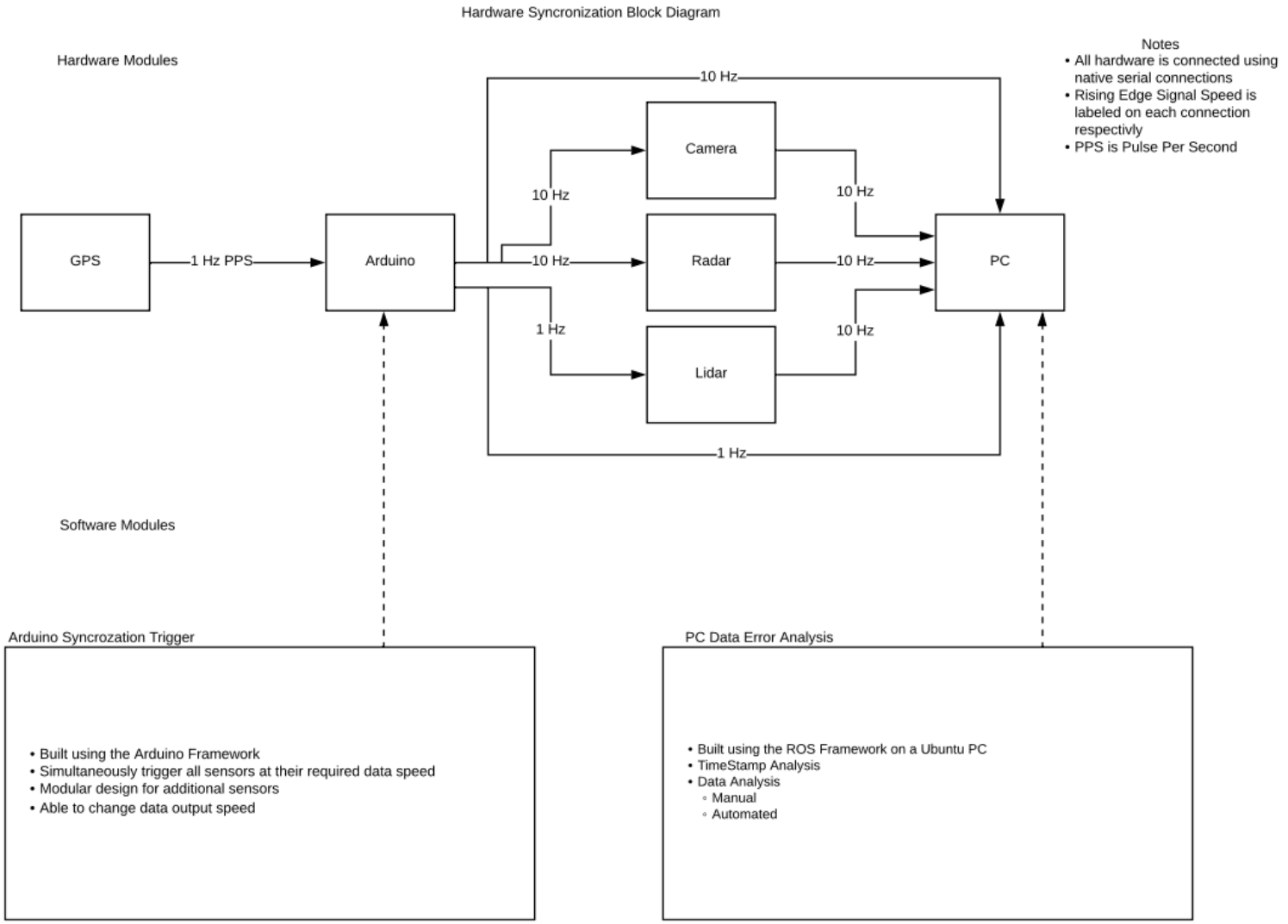


The T format is more human-readable and therefore more easily parsable, than the NGTS format. The NGTS format, however, is far more compact, so it transmits more efficiently.

3.2 Design specifications

Our design for the Hardware Synchronization is illustrated in the block diagram below. The diagram is split into two sections, the Hardware and Software. Each of the Hardware and Software modules will be described in detail, explaining what the purpose for each is and why we need it in our design.

3.2.1 Block Diagram



3.2.2 Hardware Module Descriptions

All modules will be powered and connected to any other default connections that the modules need.

GPS

The GPS has a highly reliable Atomic Clock inside of it. This Atomic Clock is what we will use to generate the Pulse Per Second (PPS) over a serial connection to the Arduino. The PPS coming from the GPS is a Rising Edge Signal that is generated at 1 Hz.

Arduino

The Arduino will take an input of the PPS from the GPS. We will have a program nested in the Arduino to generate the Outputs of a 10 Hz and 1 Hz Rising Edge Signal. These outputs will be serial connections to the sensors. The Arduino will be wired to each sensor by using the pin connectors to connect to each of the sensors' proprietary ports.

Camera

There might be multiple cameras but they will all operate the same way. The camera will take in a 10 Hz Rising Edge Signal and output the data over a serial connection at 10 Hz as well.

Radar

The radars will work very similarly to the cameras as it has one input of a 10 Hz rising Edge Signal and outputs the data over a serial connection at 10 Hz.

Lidar

The Lidar has a serial connection input of a 1 Hz Rising Edge Signal and an output of its data at 10 Hz. The Lidar is the only sensor that has an input of 1 Hz.

PC

Each of the sensors will then transmit their data to the PC over the serial connections. Each of the inputs to the PC will be at a 10 Hz rate.

3.2.3 Software Module Descriptions

Arduino Synchronization Trigger

This program will live on an Arduino in between the GPS and the sensors. The input of the Arduino will be the PPS 1 Hz Rising Edge Signal from the GPS. The output will be a 10 Hz and 1 Hz signal to the appropriate sensor and also to the PC. The Arduino will also be programmed to be modular to allow more sensors to connect to the system.

PC Data Error Analysis

The PC will receive the 10hz and 1hz signal so it can accurately use it to timestamp the data. The program will be programmed in C and will be parallelized to reduce the amount of time it takes till the data is stamped. The program will also be optimized to run as quickly as possible so that the data is time stamped as soon as possible. The PC will receive the sensor data and using the 10 and 1hz signal it will time stamp them and then it will store it in memory to be used later.

To validate the data a program will be used to calculate the error between the sensors. This program will receive the output data with the timestamps. The program will parse the timestamps then compare all the sensors at the same timestamp. An image recognition software will determine the location of a reference object. For this we will use a pendulum so that the time difference can be calculated based on the distance the pendulum is in the sensor data. Using the location of the anchor for the pendulum the program will align the images since the sensors will be viewing the object from different angles. After this the program will use the pendulum time equation to calculate the time error between the sensors. This will be run through all the data and then the average time error will be calculated to determine how accurate our hardware synchronization is.

3.3 Approach for design validation

The stated need for this project is to create a system which synchronises the data received from the sensors, so that the error of time between sensors is not above a certain value. To measure this error, there will be two different approaches that will be used to obtain this error rate.

The first way will be calculated by hand. In this case, a pendulum will be set up so that each sensor can see it. Then while the pendulum is swinging the data will be collected from the sensors and synchronized. After this, the data will be manually analyzed at the same time stamp. The data will be aligned first so that the pendulum is swinging from the same point. Then the time difference will be calculated by measuring how far apart the pendulum is from other sensors at the same timestamp. Since the pendulum is moving at the same rate the time can be calculated based on the equation of motion for the pendulum. This method will be slower, but may be used before an automated solution can be set up.

Our second method will be automated and will use the same pendulum data. However, instead of doing things by hand the positions of the pendulum will be found with image recognition software. This software will get the data points of the pendulum and the pendulum anchor as a reference point. The reference point will be used to align the images since the sensors will be in different positions. From this the program will then calculate the time delay between images and that will give us the error rate. This program will be able to go through much more data by hand, which will give us a much more accurate idea of the reliability of our hardware synchronization.

Once these methods are implemented, the error rate will be known and can be compared to the requirements. At first the data will be measured over a short period of time and then increased if the error rate is acceptable. Ing conclusion, the design validation for this project will be done in two ways. One manual and the other automatic. Before the automatic program is set up, the manual method will be used. Using a pendulum the time variation between sensors can be measured and thereby the accuracy of our hardware synchronization project.

4 Engineering standards

4.1 Project management

Qualifications

Ritwik Sathe

- Degree: BS in Computer Science with minors in Mathematics and Cybersecurity
- Relevant language experience: C++, Python
- Solid understanding of high-level networking concepts. Adept at analyzing packets using Wireshark.

Wyatt Hansen

- Computer Engineering Major and Cybersecurity Minor
- Proficient in C++, Python
- Knowledgeable in Data Structures and Algorithms

Matt Boyett

- Computer Engineering Major with a focus on computer science but a passion for hardware
- I have taken quite a few ECEN courses which deal with designing circuits,coding with low level hardware languages such a verilog, and studying how software directly interacts with hardware
- Solid level and understanding of building and diagnosing hardware systems gained from external experience

Colton Mulkey

- Extensive experience on various different software and designs
- Computer Science Major

Michael Mengistu

- Computer engineering major.
- Minor in mathematics.
- Good with finding and solving problems, testing, debugging, and hardware design.

Project Roles

Team Leader: Wyatt Hansen

Error Analysis Implementation: Colton Mulkey, Ritwik Sathe, and Wyatt Hansen

Hardware Design and Implementation: Matt Boyett, and Michael Mengistu

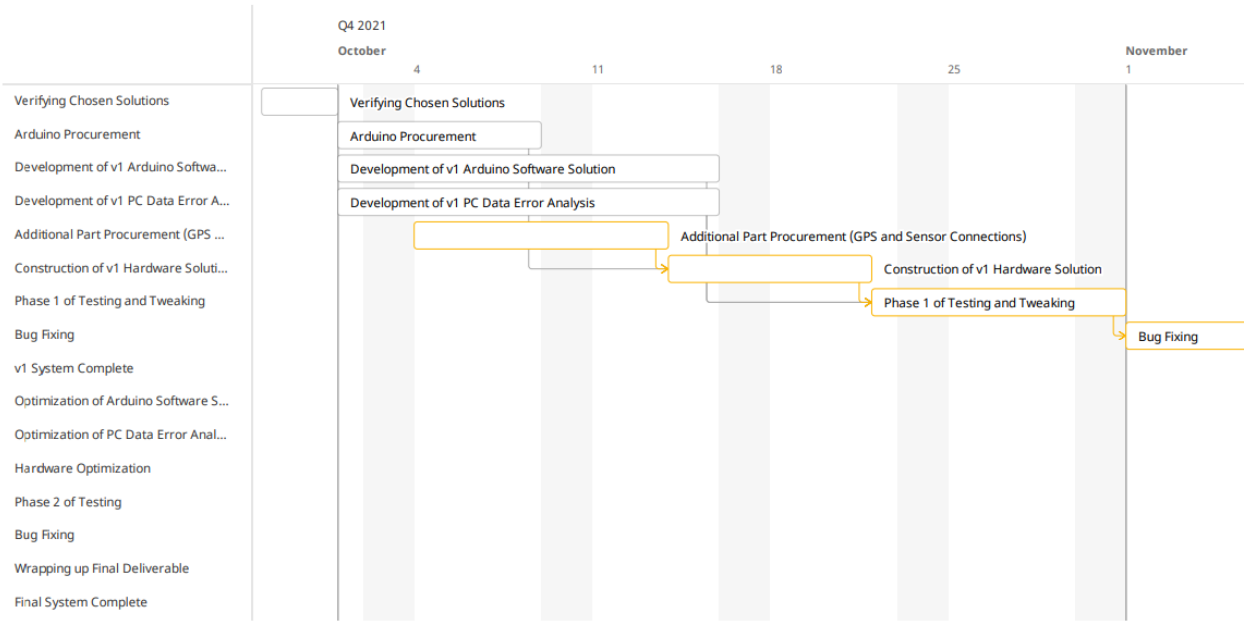
System Testing and Integration: Michael Mengistu and Ritwik Sathe

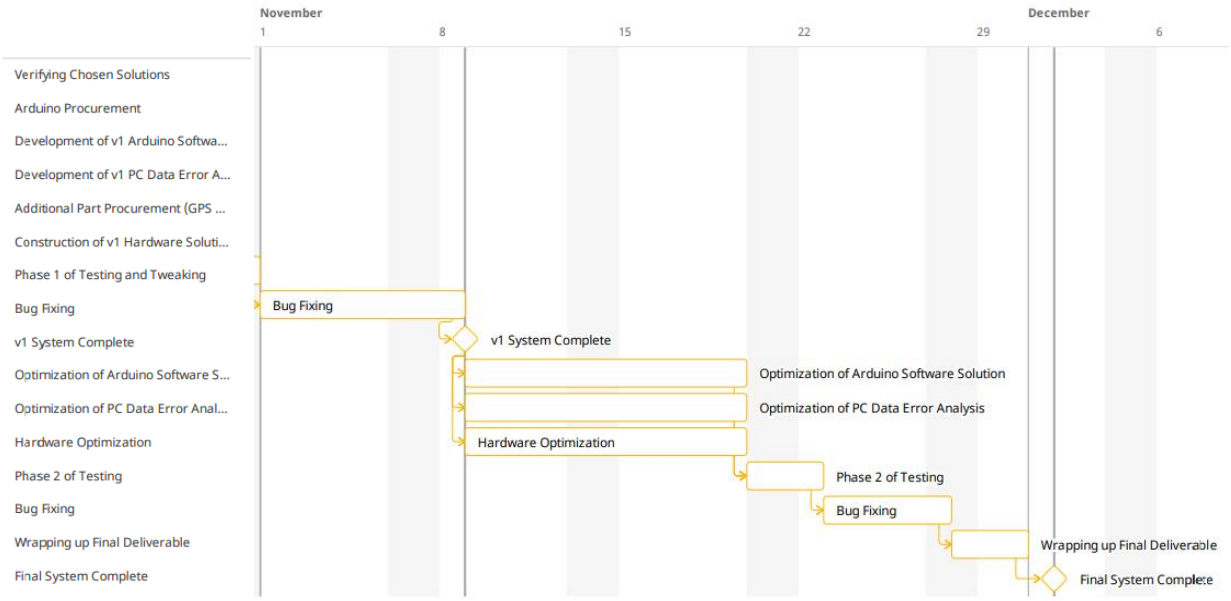
Project Management Solution

- **Gantt Chart with Critical Path**
  - Our project management solution is a Gantt Chart with a critical path method. For phase 1 of the project, both software aspects can be done concurrently without dependencies until all of the hardware pieces are obtained. The construction of the hardware side of the system, however, is dependent on obtaining all of the separate parts. Phase 1 of testing along with bug fixing is fully dependent on both the hardware and software aspects being hashed out. After phase 1, optimization of all aspects will not require dependencies but will all have the same deadline to be ready for the second round of testing. Following testing is bug fixing and wrap up of the final deliverable.
- **Brainstorming +Team Workstreams**
  - Within the allocated development periods shown on the Gantt chart, we will host designated workstream sessions at 3pm every Tuesday and Thursday to combine our collective thought processes in order to overcome any hurdles we may encounter or to just work in unison in a productive environment where we are all present.
- **Progress Checkpoints**
  - We will have weekly check-ins every Sunday at 2pm to address any problems and ensure adequate progress is being made.
  - We will also have biweekly meetings with instructors to clarify any issues and to receive any needed guidance.

4.2 Schedule of Tasks, Pert and Gantt charts

Gantt Chart Link


Q4 2021
October
November
4
11
18
25
1
Verifying Chosen Solutions
Arduino Procurement
Development of v1 Arduino Softwa...
Development of v1 PC Data Error A...
Additional Part Procurement (GPS ...
Construction of v1 Hardware Soluti...
Phase 1 of Testing and Tweaking
Bug Fixing
v1 System Complete
Optimization of Arduino Software S...
Optimization of PC Data Error Anal...
Hardware Optimization
Phase 2 of Testing
Bug Fixing
Wrapping up Final Deliverable
Final System Complete
Verifying Chosen Solutions
Arduino Procurement
Development of v1 Arduino Software Solution
Development of v1 PC Data Error Analysis
Additional Part Procurement (GPS and Sensor Connections)
Construction of v1 Hardware Solution
Phase 1 of Testing and Tweaking
Bug Fixing


November
December
1
8
15
22
29
6
Verifying Chosen Solutions
Arduino Procurement
Development of v1 Arduino Softwa...
Development of v1 PC Data Error A...
Additional Part Procurement (GPS ...
Construction of v1 Hardware Soluti...
Phase 1 of Testing and Tweaking
Bug Fixing
v1 System Complete
Optimization of Arduino Software S...
Optimization of PC Data Error Anal...
Hardware Optimization
Phase 2 of Testing
Bug Fixing
Wrapping up Final Deliverable
Final System Complete
Bug Fixing
v1 System Complete
Optimization of Arduino Software Solution
Optimization of PC Data Error Analysis
Hardware Optimization
Phase 2 of Testing
Bug Fixing
Wrapping up Final Deliverable
Final System Complete

4.3 Economic analysis

- Economical viability:
    - The potential marketability of the system can be towards autonomous vehicle companies. This is because autonomous vehicle companies rely on using the data collected from lidar, cameras, and radar sensors working together to ensure that failure from their autonomous vehicle system is not an option. As a result, focusing our marketing to companies like: Motional, Uber, AutoX, Optimus Ride, Arity, WiTricity, Unity Technologies, Ouster, Cruise, Waymo, Voyage, Swift Navigation, Embark Trucks, CARMERA, Zoox, Nauto, and Tesla, should be our objective.[3]
- Sustainability:
    - The sustainability of our system is very important because without our hardware synchronization system autonomous vehicles would not be able to accurately collect data from multiple sensors simultaneously. In order to have a sustainable system the software implementation of our hardware synchronization system must be object-oriented. This is because the code can be reusable. As a result, object-oriented programming will ensure that our system can easily accommodate different models or versions of lidar sensors from multiple vendors in case one of the lidar sensors in our system is broken for example. In addition to this, the maintenance and support of our product will require our software implementation to be able to alert the user of a sensor being broken or data not being collected from the sensors so that it can be fixed or replaced.
- Manufacturability:
    - The effect of component tolerances on our system's performance will mainly be how the hardware parts were manufactured. This is because the performance of sensors are limited by either physical or technical constraints. For example, an "ultrasonic sensor may only give a reading accurate to within a few millimeters or a servo may only be able to point to within 1 degree of its commanded position."[4]

4.4 Societal, safety and environmental analysis

The beneficial impact our project will have on society can potentially improve the reliability of autonomous vehicles. This is because the current challenge with an autonomous vehicle system is that it needs to be able to collect data; and then use the data collected to sense its environment and be able to move safely with very little or no human input. In order to do this, the autonomous vehicle system needs sensors (such as lidar, cameras, or radars to name a few) to collect data about its environment. Furthermore, the data an autonomous vehicle system collects must be synchronized. This is due to the fact that synchronization is very important in autonomous vehicles because it ensures correct sensor fusion. With correct sensor fusion, the information sourced from all the different sensors will result in less uncertainty. For example, "one could potentially obtain a more accurate location estimate of an indoor object by combining multiple data sources such as video cameras and WiFi localization signals."[1] However, there are potential downsides to our project as well. For instance, the data being collected from our sensors can potentially cause civilians to have less privacy in their daily lives. This is because of the fact that the camera sensors in autonomous vehicle systems can essentially be used to spy on individuals. Not only that, there are also safety concerns involved with our project as well. One example of a safety concern our system can potentially have could be not processing data from the sensors on time which might lead to injuries or death to pedestrians. An example of this would be an autonomous vehicle not being able to read data collected from sensors when approaching a stop sign on time. This can lead to a collision since the car cannot react until the data has been collected. Lasty, our project can potentially impact our environment in a beneficial way. One way our

project can do this is by equipping our hardware synchronization system with low cost air pollution sensors to collect real time air quality data. This data can then be used to “provide publicly available, high-quality data about air pollution, allowing many communities to gain access to information about the air around them”[2]. This is since the “main way pollution is mitigated is by management from policymakers. However, without high-quality real-time data, the policymakers don’t have a lot to go off of.”[2] Because of this, our project can show the public evidence of pollution being a problem from the data we collect.

4.5 Itemized budget

Detailed budget of all costs expected to be incurred during the project (e.g., parts, fabrication services).

Hardware and Cost		
Hardware	**Amount**	**Cost($)**
Arduino	1	23
GPS	1	~20
Camera	4	~20
Radar	1	~50
Lidar	2	~100
Connectors	20	~7
Total Cost		393

Hours		
Name	**Hours worked**	**Expected Pay Per Hour**
Michael mengistu	100	20
Matt Boyett	100	20
Ritwik Sathe	100	20
Wyatt Hansen	100	20
Colton Mulkey	100	20

5 References

[1] “Sensor fusion,” Wikipedia, 30-Jun-2021. [Online]. Available: https://en.wikipedia.org/wiki/Sensor_fusion. [Accessed: 26-Sep-2021].

[2] M. Brewer, “Low-cost sensors to fight air pollution,” College of Engineering at Carnegie Mellon University. [Online]. Available: https://engineering.cmu.edu/news-events/news/2020/10/06-sensors-air-pollution.html. [Accessed: 26-Sep-2021].

[3] S. Betz, “The top 25 self-driving car companies paving the way for an autonomous future,” Built In. [Online]. Available: https://builtin.com/transportation-tech/self-driving-car-companies. [Accessed: 26-Sep-2021].

[4] “TOLERANCE ANALYSIS,” *The Grainger College of Engineering*. [Online]. Available: https://courses.engr.illinois.edu/ece445/documents/tolerance-analysis-guide.pdf. [Accessed: 26-Sep-2021].

[5] M. Faizullin, A. Kornilova, and G. Ferrer, “Open-Source LiDAR Time Synchronization System by Mimicking GPS-clock,” arxiv.org. [Online]. Available: https://arxiv.org/pdf/2107.02625.pdf.

[6] T. Flir, “Using PPS to Synchronize with External GPS,” *Teledyne FLIR*, 13-Jul-2013. [Online]. Available: https://www.flir.com/support-center/iis/machine-vision/application-note/using-pps-to-synchronize-with-external-gps/. [Accessed: 26-Sep-2021].

[7] K. Y. Koo, D. Hester, and S. Kim, “Time synchronization for wireless sensors using low-cost GPS module and Arduino,” *Frontiers*, 01-Jan-1AD. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fbuil.2018.00082/full. [Accessed: 26-Sep-2021].

[8] M. Desai and M. Upadhyay, “Generation of GPS Based Time Signal Outputs for Time Synchronization Application,” *International Journal of Advance Engineering and Research Development*, vol. 3, no. 4, pp. 910–914, Apr. 2014.

[9] S. Liu, B. Yu, Y. Liu, K. Zhang, Y. Qiao, T. Y. Li, J. Tang, and Y. Zhu, “Brief industry paper: The matter of time — a general and efficient system for precise sensor synchronization in robotic computing,” *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Mar. 2021.

[10] R. Teja, “What are the differences between Raspberry Pi and ARDUINO?,” *Electronics Hub*, 05-Apr-2021. [Online]. Available: https://www.electronicshub.org/raspberry-pi-vs-arduino/#Differences_between_Raspberry_Pi_and_Arduino. [Accessed: 26-Sep-2021].

6 Appendices

6.1 Product datasheets

Arduino DataSheet: https://www.arduino.cc/en/uploads/Tutorial/595datasheet.pdf