AR Monitoring System

Team Number: Dec1714
Client: Andrew Guillemette
Advisor: Goce Trajcevski
Arbaaz: Team Leader
Ben: Key Idea Holder
Sam: Key Idea Holder
Nipun: Communication
Patrick: Webmaster
Dheeraj: Communication
Dec1714@iastate.edu
Dec1714.sd.ece.iastate.edu

Dec. 5, 2017

Introduction

The AR monitoring system aims to use augmented reality technology to further improve oversight in a worksite. Coupled with tracking hardware, our AR monitoring system can track the location and speed of any vehicle on at the worksite. Throughout this semester we continued our work on software by improving the hololens app and setting up our own server; while making a switch in our hardware focus from camera/gimbal motion to reading CAN bus information and real time gps tracking.

1.1 Project Statement

As society has grown the ability to use technology to simplify everyday-processes has likewise increased. With the use of Augmented Reality (AR) we can greatly improve the ability of companies to monitor vehicle operations at remote worksites. The intention of this project was to design an AR monitoring solution that would be integrated into the cab of a worksite vehicle such as a truck, bulldozer, or crane in order to monitor the usage of the vehicle. At the other end a user or users would use a Microsoft Hololens which would generate an aerial map view of the worksite displaying virtual 3D models of the worksite vehicles moving throughout the map in real time. The user would then be able to select any vehicle fitted with the monitoring hardware and access vehicle-specific diagnostic information. This would allow modifications to be made to improve the workflow and efficiency on the worksite.

1.2 Purpose

The purpose of this project is to develop an AR monitoring solution in order to remotely track and monitor worksite vehicle activities. This solution aims to provide a convenient means by which a supervisor may remotely “view” a worksite and its current vehicle operations via an AR headset and also be able to acquire real-time vehicle diagnostic and GPS information.

1.3 Goals

The end goal for this project was to have a complete set of software and hardware necessary to remotely monitor a wide array of machinery as a proof of concept. This overarching goal can be broken into two primary components: that concerning the monitored machine and that of the remote Hololens end-user. With regards to the machinery-end, inherent to our goal is the collection and relaying of vehicle diagnostic information. The core of our goal on the Hololens end-user side is the ability to visualize in real-time a remote worksite including all of the monitored vehicles with the ability of acquiring current vehicle-specific information via communication with a remote server.

2 Deliverables

The deliverables for our project changed at the start of the second semester. After the first semester our client decided to prioritize the CAN Bus and GPS functionalities over that of gimbal and camera. The hardware team spent much time during the first semester on developing a gimbal/camera unit to facilitate users of the HoloLens application having a first-person “view” of the inside of the vehicle cabin. The original idea was to have the camera mounted onto the gimbal and to have the gimbal pivot based upon the orientation of the HoloLens headset. It was decided that the gimbal and camera functionality was no longer a primary interest of our client. Once the second semester began the decision was made to drop the gimbal/camera functionality from the project and to focus on acquiring real-time GPS coordinates and CAN Bus diagnostic information. Once we shifted our focus we then went into setting up the new deliverables for our hardware team.

The software team’s deliverables changed at the start of the second semester due to a complete redesign of our server. Since we shifted from our custom server application to Redis, we needed to write a custom library that could provide a consistent API that applications on both the Hololens and Raspberry Pi could use to communicate with each other.

2.1 Hardware

- Make requests for and receive information from CAN Bus pertaining to speed, engine temperature, RPM, throttle, faults, and misfires
- Use GPS receiver to acquire GPS coordinates
- All code runs when system starts
- Send acquired GPS and CAN Bus data to server
- RFID Authentication

2.2 Software

- Write a custom library to interface with redis server via the raspberry pi and hololens
- Set up handlers to listen to CAN Bus and GPS data via their respective channels
- Project 2D scrollable map on a table through the hololens
- Move virtual vehicles on the map in real time using GPS redis channel
- Create data visualization table and display it with real time CAN Bus data when a vehicle is clicked on.

3 Design

In our design we had two different sets of requirements. Functional requirements, the requirements that describe what the system should actually do, and nonfunctional requirements, the requirements which specify how the system should accomplish what it is intended to do.

3.1 System Specification

The specifications for this project are outlined in the deliverables section. Our goal for this semester of the project is to have an independently established server that allows communication for Raspberry Pi devices to a hololens device. The specification that we have identified revolves around the functionality of the Raspberry PI, stability of the server and functionality of the hololens. The Raspberry Pi functionality revolves around being abe run our programs when it starts up and is able to communicate with the server. Stability of our server is for it to be able to handle communications from various devices. Finally, the specifications for hololens functionality are for to properly display a site location with moving 3D models for vehicles.

3.1.1 NonFunctional

- The software should be able to be adapted for many different purposes and inputs.
- The software should be secure. Unauthorized devices should not be able to feed false data nor should they be able to view device data.

3.1.2 Functional

- Ability to view monitored machine on map using Hololens app
- Ability to view a custom area map on the Hololens app
- Ability to assess machine status from Hololens app
- Ability to monitor desired CAN bus output from the Hololens app

3.2 Proposed Design

The design of the system will be split into three main components: the remote device which will be located in the vehicles being tracked, the AR application which will run on a Hololens, and a server application which acts as an intermediary to relay communication between remote devices and the AR application,

The remote device is based on a Raspberry Pi. It has a GPS module to determine its location, a module to connect to the vehicle’s CAN Bus network and read its diagnostic data, and a module that connects to the server wirelessly. The software that runs on the Pi controls all these devices, acquires the necessary information, and sends the information to the server over the network.

The server will be the backbone of communications between all the devices in the system. Since having devices detect each other among a sea of devices or perhaps behind a firewall isn’t optimal, this is the best solution. Acting as a bridge between the observer and the observee, there must be a degree of differentiation between the two. Because the observed devices are located in remote locations that may not be physically secure, these devices should have limited access to the system and should only “offer” data to the server. The server should only consume the data for the purposes of passing it onto the HoloLens. The server uses redis as a database and as a message queue to implement real-time communication among client devices.

The HoloLens application will be set up using Unity. Unity offers the best support for the HoloLens being the officially recommended engine upon release. Installing third-party software along with integrated versioning makes it an ideal candidate for development. Along the lines of third-party software, some excellent sub-solutions for features are only available via Unity. Using the engine, we will display a map on a physical surface and use it to pinpoint the locations of various remote devices, and display any relevant data when the vehicles are selected.

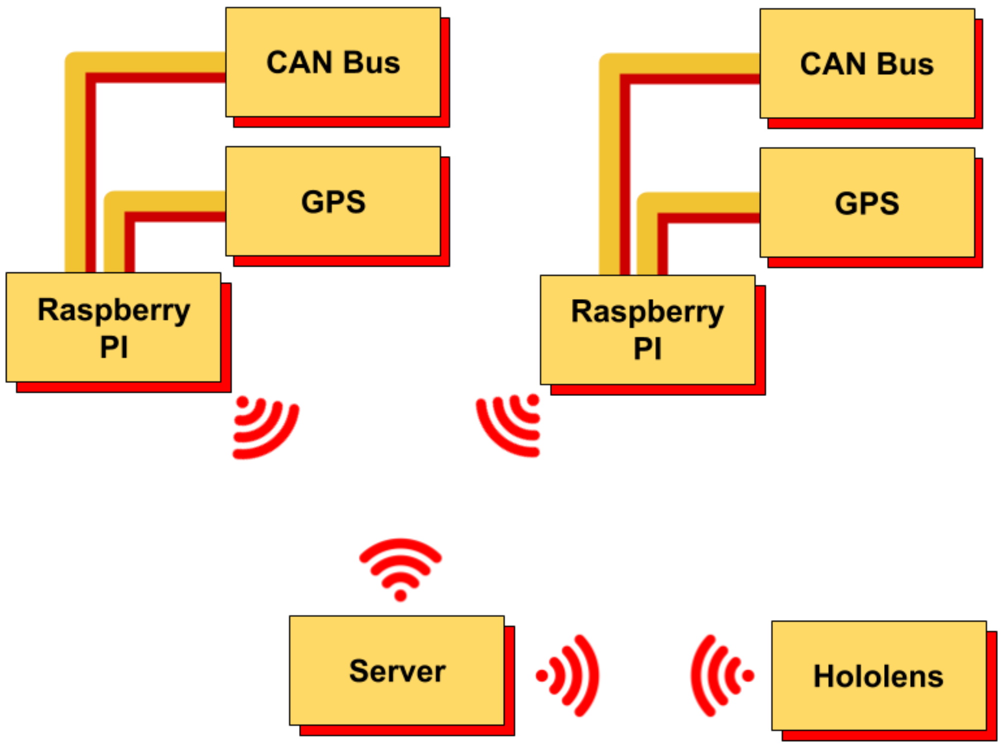

Figure 1 - Block diagram detailing an overall system perspective of the final product

4 Testing

4.1 Interface Specifications

All communications between devices are to be sent through the central server. Each class of device has its own sub-service that it communicates with. The purpose of these subservices are to separate the roles of devices and to establish a reasonable protocol for communication. For example, remote devices that are out in the field or travel long distances may not be expected to have a continuous connection with the server and should have an intermediary service to step in and represent the device on its behalf. On the other hand, monitoring devices have a typical use case of being in a corporate headquarters or some other well established location with a readily available connection. The intent of these subservices are to alter the way communications to and from a device are handled but not to change the communications themselves. The exact language used to describe the state of a device is arbitrary but we did define one. While we are unable to provide a full description, a loose analogy can be used comparing it the reads and writes of files in a folder system. The only absolute requirement of any device wishing to communicate with the server is that it is able to read and write its message to the channel of communication set aside for the remote device that is being described by the message.

4.2 Hardware

Testing done on the hardware components focused on functionality and completion of our deliverables. The two main components that we tested were the GPS receiver and the CAN Bus reader. For both of those aspects we focused on developing code that would properly acquire needed information for our system and relay that information to the remote server. The testing of our components were broken down into their respective parts, once the testing of our separate components was successful we then moved into testing our hardware system as a whole.

4.3 Software

Software testing was centered on three different areas. Scripts were written on the HoloLens side to be run within the Unity editor. The editor provided a convenient GUI interface to click and run tests and miscellaneous code. We also provided sample data to the server to pass on to the HoloLens for processing. With the primary purpose of the Hololens being visualization, the testing was visually verified for correctness. Server testing was minimal due to the nature of our communications design. The server is unaware of the technicalities of the communications it is sending and receiving. Therefore, we connected to the server with a basic client and sent dummy data to verify that it was indeed passing forth the data it received. Testing on our Raspberry Pis proved to be a little more tricky. These devices had limited processing power to reasonably perform rapid testing on variations of our code. The base logic had to be tested on a development machine and the whole package was simply tested for crashes on the sensor device.

5 Results

5.1 Hardware

We began testing the functionality of our GPS receiver and CAN Bus by using test scripts, once we had determined that these parts were working we then went into writing our own program to operate them. Our program for GPS coordinates is accurate within 10 meters of the receiver location. The CAN Bus reader is properly pulling the speed, throttle, and rpm information from any car that we plug our device into. This programs are able to successfully run when our system turned on and is able to stop properly when our system is turned off. Testing the RFID proved to be a greater challenge than could be accomplished within our time constraints, thus we determined it would be best to have it cut from our feature list so that we can manage to ensure that the rest of our devices were operating properly. When testing communication from the Raspberry Pi to the server we found that we can successfully ping the server, however there were issues in establishing proper communication with the two.

We had decided to take a risk by increasing development costs in the immediate future in the hopes of saving time in the long run. Understanding that our systems were prototypes, we chose to write our client code in a manner that was platform independent. The code included a modular framework for minimal maintenance costs if the server system was ever redesigned. Our prototyping code used on the Raspberry Pi was restricted to Python for technical limitations of other languages and their interfaces with the attached sensors. In a fully completed product, our hardware may be something a little more lean and incapable of such an abstract language. We decided to write our client framework in C++ and used a third-party tool to provide us with an interface to many common languages including Python. Due to differences in the way the two languages handle data, the required time to debug exceeded our budgeted time. Despite being incomplete, the functionality of the client framework was able to be tested in its native language and was verified to be processing data as expected.

5.2 Software

The HoloLens has been successfully parsing data received from the server and is able to manipulate the virtual environment based on the data. Most of the effects are still untested as they are under development. Server was tested to be successful by registering two simple clients to the server. We were then able to successfully pass messages between them. The current software used on the sensor units is currently untested as part of it is still under development. The completed portion is part of a language port from HoloLens, which has already been tested to be working.

6 Conclusion

Though the final implementation of our AR monitoring system was not able to be brought to a perfect state of completion, our project does serve as an effective proof of concept for using AR as a beneficial solution for monitoring worksite vehicle activities. We can successfully acquire both vehicle diagnostic and GPS information and prepare the data to be transmitted to a remote server. We have the ability to pull information from the remote server and use it in a HoloLens application in order to display worksite vehicle activities in real-time on a custom map of a worksite. A supervisor could use this application to monitor the vehicle activities on their worksite of interest without ever having to actually set foot on the physical worksite. The breakdown in our final implementation was specific to the custom libraries written to manage and transmit the data collected on the remote vehicle. If this issue was resolved then we would be able to stream data from the remote vehicles to the HoloLens application by using the server as the intermediary.

The solution which we have proposed in this report is by no means a product without room for innovation and improvement. There are many ways by which this monitoring solution could be refined and further developed. One way to further develop this project would be to incorporate the use of the gimbal/camera module which we had been working on implementing during the first semester. This would enable supervisors to experience a first-person view of the inside of worksite vehicles remotely. Another possible feature to include in this project is that of personnel identification via RFID tags. Using this RFID system would allow logging of miles and hours put in by personnel in each vehicle. A final improvement to hardware to be made would be improve the GPS accuracy by using Global Navigation Satellite System.

7 Appendix

7.1 Operational Manual

This section outlines how to use each respective part of this project. Note that it is assumed that all necessary software installations have been completed previously. It is also assumed that the necessary additional hardware components such as the CAN Bus shield for the Raspberry Pi are already in there proper positions and are ready for use.

Instructions for the Remote Vehicle User

1. Connect all cables and peripheral devices to Raspberry PI board. The list of cables and peripherals are as follows:
    a. GPS Receiver (Plug into one of the four available USB ports)
    b. OBD2 cable (for CAN Bus communications)
    c. Power cable for Raspberry Pi (USB to micro USB)
2. Connect the other end of the OBD2 cable into the CAN output port which is usually beneath the steering wheel
3. Make sure to connect the Raspberry Pi's power cable to a power adapter in the vehicle
4. Turn on the vehicle into full operational mode. Data will automatically be collected and transmitted to the remote server

Instructions for the HoloLens User

1. Open the LTEObserver App within the HoloLens Device.
2. When prompted, look at a surface and tap to place a map.
3. Markers indicating device locations will be displayed on the map.
4. Click on the map and drag to move the map around.
5. Click on the magnification icons displayed above the map to zoom in or out.
6. Click on the device markers to show additional information about the device.

7.2 Alternative/ Initial Design

Over the course of the two semesters we have had several alternative designs that either did not come to fruition or were put aside in order to work on a different part of the design. This included three different versions of the in cab hardware setup and another 2 different designs for our service architecture.

In the first semester the hardware setup was mostly concentrated on the design and implementation of an in cab camera setup that would allow the user on the Hololens side to have a first person view of the cab. This FPV camera would be attached to a gimbal controlled by the movement of the Hololens operator's head. This design was put on the waiting list in the second semester due to the our clients desire for the entirety of the CAN bus operation to be in order by the end of this semester. With some extra time predicted for the end of the semester there was also an attempt to implement RFID to log vehicle usage. This was ultimately scrapped due to lack of time leaving us with our current model.

There was also a plan to build video streaming support into the server and support for live playback of the stream into the Hololens app. This was supposed to work with the in-cab camera and gimbal setup mentioned above, but it was scrapped from the project after the first semester. Time constraints also meant that we could not add user authentication to our server, which was a part of our plan during the first semester.

7.3 Related Material

It has been shown that AR (Augmented Reality) can also be used to keep record of construction site progress (Zollmann). Zollmann, Hoppe, Kluckner, Poglitsch, Bischof, and Reitmayr have shown in their research publication *Augmented Reality for Construction Site Monitoring and Documentation* “how to use AR to support monitoring and documentation of construction site progress” (Zollmann).

It is proposed that using AR would address many issues inherent to construction site progress monitoring. In order to track construction site progress today oftentimes digital photography will be used by a surveyor on the construction site. The problem with this approach is that it is both time consuming and imperfect (Zollmann). With the help of AR it is shown that automated documentation of the worksite is possible by using an aerial client to gather a 3D reconstruction of the worksite. By using a more-encompassing method of documenting the progress of a construction site it is further reasoned that contractor’s achievements can be better assessed. This improved monitoring would also enable supervisors to better trace issues and defects concerning the progress of the construction site to their source.

In order to accomplish this construction site monitoring project the researchers “developed a system consisting of three main components: an aerial client that captures aerial images for 3-D reconstruction on regular basis, a reconstruction client that performs aerial reconstructions and remote localization, and, finally, an AR client that visualizes progress information spatially registered to the physical environment on site. Each component is able to communicate with the others and to exchange data over a network” (Zollmann).

What is discussed here is just a high-level view of this particular project, but a more information about the project can be gained by referring from the actual research paper.

This construction site progress-monitoring project is of interest to our own project because is it very similar to our own project in that AR technology is being used for the purpose of worksite monitoring. The key difference between the projects is that this related project is concerned with specifically monitoring the progression of development of the construction site whereas our team is concerned with the monitoring of worksite vehicles. Both projects aim to simplify and improve the management of construction site projects.

There is the potential for an integrated solution of both projects combined: the monitoring of the actual development of construction sites paired with the monitoring of the vehicles on the worksites. Both projects address different aspects of the construction process and if integrated together and further developed could provide for a more holistic monitoring solution for construction sites.

7.4 References

Zollmann, S., Hoppe, C., Kluckner, S., Poglitsch, C., Bischof, H., Reitmayr, H. (2014). *Augmented Reality for Construction Site Monitoring and Documentation*. New York City, NY: IEEE.