

UNIVERSITY OF WESTERN AUSTRALIA  
SCHOOL OF ELECTRICAL, ELECTRONIC AND  
COMPUTER ENGINEERING

**Improved Localization in UWA REV Autonomous  
Driving SAE Vehicle Through IMU, GPS, Wheel  
Encoders and Extended Kalman Filter**

---

**Mitchell Poole**

**Student Number: 21212271**

**Supervisor: Dr. Thomas Bräunl**

**School of Electrical, Electronic, and Computer Engineering**

This thesis is submitted for the  
degree of Master of Electrical &  
Electronic Engineering

Final Year Project Thesis

The University of Western Australia

Submitted: 30<sup>th</sup> October 2017



THE UNIVERSITY OF  
WESTERN AUSTRALIA  
*Achieve International Excellence*



## Abstract

The following dissertation describes the improvements developed for the localization system of the Autonomous SAE vehicle at UWA. The main improvements that were applied to the localization system are the addition of odometry to the low level control system, as well as an Extended Kalman Filter, and changes to the low level software, and to the sensor fusion algorithms.

Odometry sensing to the SAE vehicle was added in by use of a microcontroller and Hall Effect Sensors. This microcontroller communicated with the microcontroller in charge of steering, braking and acceleration of the vehicle, to provide wheel speed measurements to the main microcontroller into the main processing system. The sensor fusion algorithm attempted to compare and contrast at least three cases; sensor fusion with odometry with a Kalman filter, sensor fusion with odometry using a Kalman Filter, and sensor fusion with odometry and an EKF.

A reliable odometry measurement sensor has been obtained, that provides speed for each of the four wheels. The wheel speeds are then captured by the central control unit (An NVidia Jetson TX1 board), which uses the wheel speeds for localization in its control system. Minor calibration will be needed for more accurate wheel speed measurements, outside of this, the odometry is a partially integrated system in the SAE vehicle. The Kalman filter sensor fusion variants and the EKF could not be tested on due to errors in the fusion class, which needs to full fully integrated into the system.

The addition of the Jetson board allows others to improve the localization of the SAE vehicle by adding in some form of monte-carlo algorithms, such as a Particle Filter, or using an Unscented Kalman Filter for localization. The improvements to localization allows for the implementation of a SLAM type control system for the SAE vehicle, or other control system features.

## Acknowledgements

Dr. Thomas Braunl for his leadership and vision over this year, as well as providing me with the opportunity to work on the REV.

The REV team, for their teamwork and camaraderie throughout the year.

My friends and family for their unconditional support over this year.

Altronics for their continued sponsorship of the UWA REV project.

## Nomenclature

CLI – Command Line Interface

EKF – Extended Kalman Filter

GPS- Global Positioning System

IMU – Inertial Measurement Unit

ISR – Interrupt Service Routine

RTK-GPS – Real Time Kinematics GPS

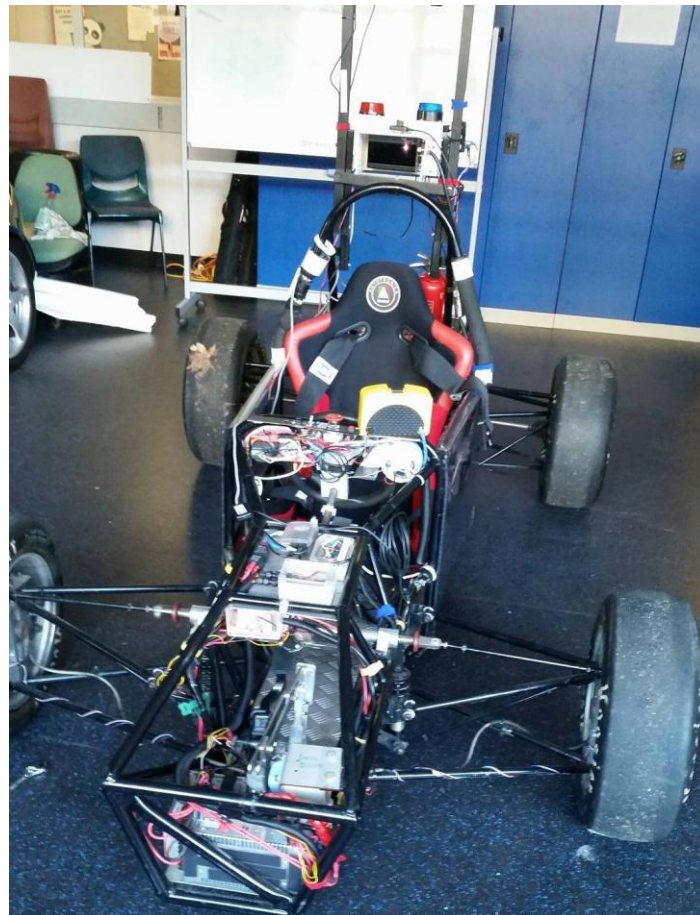
USB – Universal Serial Bus

|  |    |
|--|----|
| Abstract.....  | 1  |
| Acknowledgements.....  | 2  |
| Nomenclature.....  | 2  |
| 1. Introduction and Background.....                            | 4  |
| 2. Literature Review.....                                      | 5  |
| 3. System Design .....   | 6  |
| 3.1. Hardware.....   | 6  |
| 3.2. Software.....   | 6  |
| 3.3. GPS .....   | 7  |
| 3.3. IMU.....  | 8  |
| 3.4. Cameras.....  | 9  |
| 3.5. Lidar .....   | 9  |
| 3.6. Odometry .....  | 9  |
| 3.7. Ackerman Drive Model .....                                | 11 |
| 3.8 Low Level Improvements .....                               | 11 |
| 3.8.1 Formatting.....  | 12 |
| 3.9.1 Kalman Filter .....                                      | 12 |
| 3.9.2 Extended Kalman Filter .....                             | 12 |
| 3.9.3Sensor Fusion.....  | 13 |
| 4. Results.....  | 14 |
| 4.1. Odometry .....  | 14 |
| 4.2. Low Level System Changes.....                             | 16 |
| 4.3. Kalman Filters.....                                       | 16 |
| 4.4. EKF.....  | 16 |
| 5. Conclusion .....  | 17 |
| 6. Future Work.....  | 17 |
| 7. References.....   | 18 |
| 8. Appendices.....   | 20 |
| Appendix A - Low level circuit changes .....                   | 20 |
| Appendix B – Odometry and Opto Coupler Isolation circuit ..... | 21 |
| Appendix C – Low Level Code Changes.....                       | 21 |
| Appendix D – Fusion code changes.....                          | 27 |
| Appendix E – Low_Level_Timer.ino .....                         | 33 |
| Appendix F – Low_Level_Odometry.ino .....                      | 47 |
| Appendix G – List of Software Used.....                        | 51 |

## 1. Introduction and Background

There has been significant interest in autonomous driving over the past few years, with car manufacturers and semiconductor manufacturers attempting to enter this developing market, such as Tesla, Google, and Toyota. An early example of Autonomous driving can be traced back to 1925, in which a radio controlled car in the streets of Fredericksburg, Virginia was demonstrated [1]. Since this time though, the continuing progress of this area has advanced to such a degree that Google has successfully developed an autonomous car that has driven approximately 300,000 miles since 2012 without an accident [2]. This field has yet to take off commercially, but with Toyota has announcing plans to release a commercial autonomous vehicle by 2020 [3], it may not be long until autonomous cars become a regular addition to modern life. Before this happens, Improvements should be made to the standards of this field before it can take off in the market.

Participating in this emerging field, the UWA Renewable Energy Vehicle (REV) Project has made an Autonomous SAE vehicle as a research platform. This car started off as running on an internal combustion engine, but was converted to an electric car in (08/09), and then an additional drive-by-wire system was added in (11/13). In 2017, the goal was to get autonomous driving with cone detection and path planning that emulates the demonstration shown in the Formula Student Germany [4].



**Figure 1: The UWA Autonomous SAE Vehicle**

This project was carried out in a team with four other students; Gabriel, Roman, Sam and Jason with two PhD students; Thomas and Kai. For object detection and mapping, Gabriel, Roman and Sam were in charge of using the Lidar to find objects, as well as the control and path planning. Jason and Kai were doing the computer vision and object detection of the SAW Vehicle, and Thomas and I were in charge of the localization.

## 2. Literature Review

The history of the SAE vehicle is that it started off as a internal combustion engine powered SAE car, that was converted to an electric vehicle in 2010 [5] and then autonomous driving and a drive-by-wire system was implemented in 2013 [6]. In 2015, the software was updated in to refine the path planning algorithm by using an Extended Kalman Filter (EK-Filter) over a regular Kalman Filter as well as implementing RTK-GPS module [7]. The RTK-GPS module was done using a component from a company called Swift Navigation called a Piksi GPS. From this the SAE vehicle control software was changed to pick between the RTK-GPS and normal GPS device. The results of the work performed in [7] was improved localization due to the RTK-GPS module, though the EKF required more work and possible fusion with vehicular odometry to drive effectively in the absence of a stable GPS connection. Investigation into the odometry of the SAE vehicle found that it did not work, and thus needed to be re-implemented.

The system was centralized around a Raspberry Pi as the main processor, with all the sensors sending it their data so it can make decisions based on this. While the Raspberry Pi was used in [5] and [6], and [7] utilized a Raspberry Pi 2, it was too slow in which it ended up hampering progress. So the system was moved to a Raspberry Pi 3 in 2016/2017 to provide greater processing power.

Odometry was achieved originally by using the Arduino UNO that controlled the steering, braking and acceleration to record the speed of each wheel of the vehicle. Doing all these tasks proved to be nonviable for the microcontroller, as the system did not possess the processing power to be able to handle all four of these tasks, especially due to how the odometry was implemented in code, which used an interrupt service routine, which had to share its time with other ISRs in the low level software. To resolve this issue, the low level control circuit and the circuit for the odometry were recommended to be separated for modularity, and an additional microcontroller or another method to perform the odometry is to be added.

Extended Kalman Filters have been widely implemented since the use and are the most widely used filter on localization in the world [8]. Many studies employ the use of Extended Kalman Filters as a tool or a testing system for localization. A comparison of Kalman filter variants is presented in [9], including an unscented Kalman filter, extended Kalman filter and two  $H_\infty$  filters, each compared for their performance. The study found that the unscented Kalman filter and the EKF had similar performance, as long as the non linear elements of the system are non severe, the EKF and the Unscented Kalman Filter have similar performance, but when the non linear elements are severe, the Unscented Kalman Filter had superior performance, in addition to no significant improvement in performance when comparing the Kalman filter variants to the  $H_\infty$  variants. This conclusion was also reached in [8], as well as noting that implementing the Unscented Kalman Filter was easier than the EKF.

In [10], a system of sensor fusion involving multiple sensors with a Kalman filter is proposed and tested on a radar tracking system. The sensor fusion algorithm follows a complex system of two layered Kalman Filtering and is on a radar system that uses position, velocity and acceleration in the state transition matrix. The result of this mutli-sensor mutli-layered Kalman Filter sensor fusion algorithm was it was optimal for systems with multiple sensors and correlated noises. As our position, velocity and acceleration are needed to get results, and these noises are likely not correlated, this studies' sensor fusion model would not prove useful to the project.

There are papers with numerous ways of sensor fusion with different types of sensors. Sensor fusion algorithms with various combinations of sensors including GPS, IMU and Wheel Encoders along with other sensors are performed in [11], [12] and [13]. The robot in [11] had five sensors, including an

encoder, compass, GPS, gyroscope and accelerometer, which were all fused with an EKF. The study used its own variation of the state transition matrix and the measurement matrices for each of its sensors. The sensor fusion involved fusion of both of the position, velocity, acceleration and headings.

In [12], a sensor fusion algorithm employing the EKF is employed and the algorithm employs the use of force body dynamics for use in the modelling the state transition vectors of the EKF.

In particular [13] described a robot which used IMU and Wheel Encoders, as well as image processing and utilizing a Kalman Filter for sensor fusion. The IMU was stressed as low cost due to utilizing a Nintendo Wii remote as an improvised IMU, at estimated \$42USD at the time of writing. The result of the sensor fusion from [13] was a drive that was very close to the actual path, which was compared to drives with only one of each sensor. All three of these studies provide useful material to setting the state transition and observation matrices for making an EKF to suit our purposes.

## 3. System Design

### 3.1. Hardware

Autonomous driving systems are dependent on relatively powerful microprocessors, depending on the sensors used. Initially the SAE vehicle used a Raspberry Pi for this application. As the system is being implemented with cameras, and an IBEO LUX Light Detection and Ranging Laser Scanner, for object and feature detection, a more robust, device is needed to accommodate the processing power needed from all of these scanners. Comparing the Raspberry Pi 3's specifications, featuring a 1.2GHz ARM processor, 1GB LPDDR2 RAM, and 16GB Micro SD card storage [16], with a NVidia Jetson TX1 board, which uses a Quad ARM® A57/2 MB L2 CPU , 4GB 64 bit LPDDR4 RAM, and features various other types of data storage [14]. For this reason, the Raspberry Pi was swapped for the NVidia Jetson TX1 board, which allows the SAE vehicle to become a more powerful autonomous driving system.

A significant redesign of the previous setup of the autonomous SAE vehicle has taken place in order to improve the car's functionality. An additional DCDC converter from Mean Well [15] to convert the LiPo battery output voltage from 48V to 12V to power the NVidia Jetson TX1 board was utilized. To provide power to a 10 port USB hub, a 12V to 5V DCDC converter was also utilized. Initially the Lidar was connected to the Raspberry Pi via an Ethernet modem, but after replacing the Raspberry Pi with the NVidia Jetson TX1 board, the modem was able to be removed from that, and the Lidar was directly inputted into the NVidia JetsonTX1 board through its own cable into the Ethernet socket of the NVidia Jetson TX1 board.

### 3.2. Software

The Software architecture of the project, from [5] is centralized around the control class, with wrapper classes for each of the sensors, as well as the fusion class. The software includes a web based framework by [5], which allows for a simpler use than a CLI. From there the output of many variables can be displayed, which will be employed significantly, as well as the Logger class, which record all the information needed.

### 3.3. GPS

From [7], a RTK GPS system was tested for the localization of the SAE vehicle. This proved to be very useful for localization, when the system was reliable. Currently the GPS module being used is a Columbus V-800 USB GPS [18]. Besides being able to attain latitude and longitude from the system, the GPS can also be used to obtain track angle, velocity and acceleration, though since the V-800 USB GPS module is not an RTK-GPS module, these measurements will be poorer than it could be. This proves to be especially true when these measurements are found through integration.

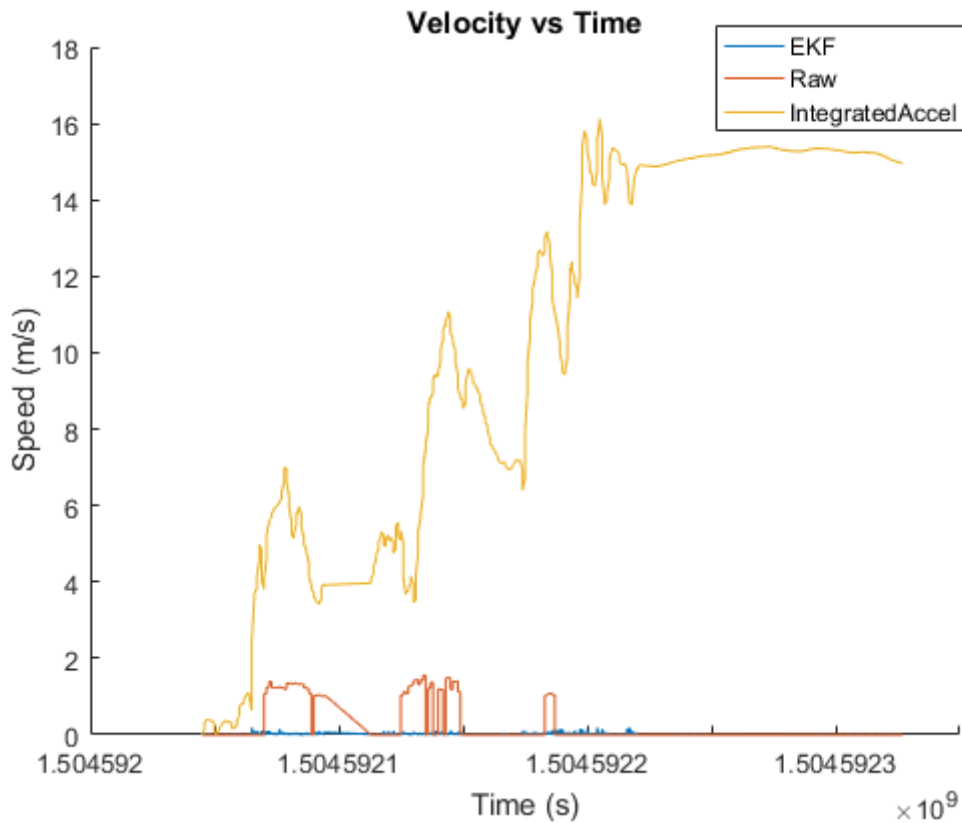
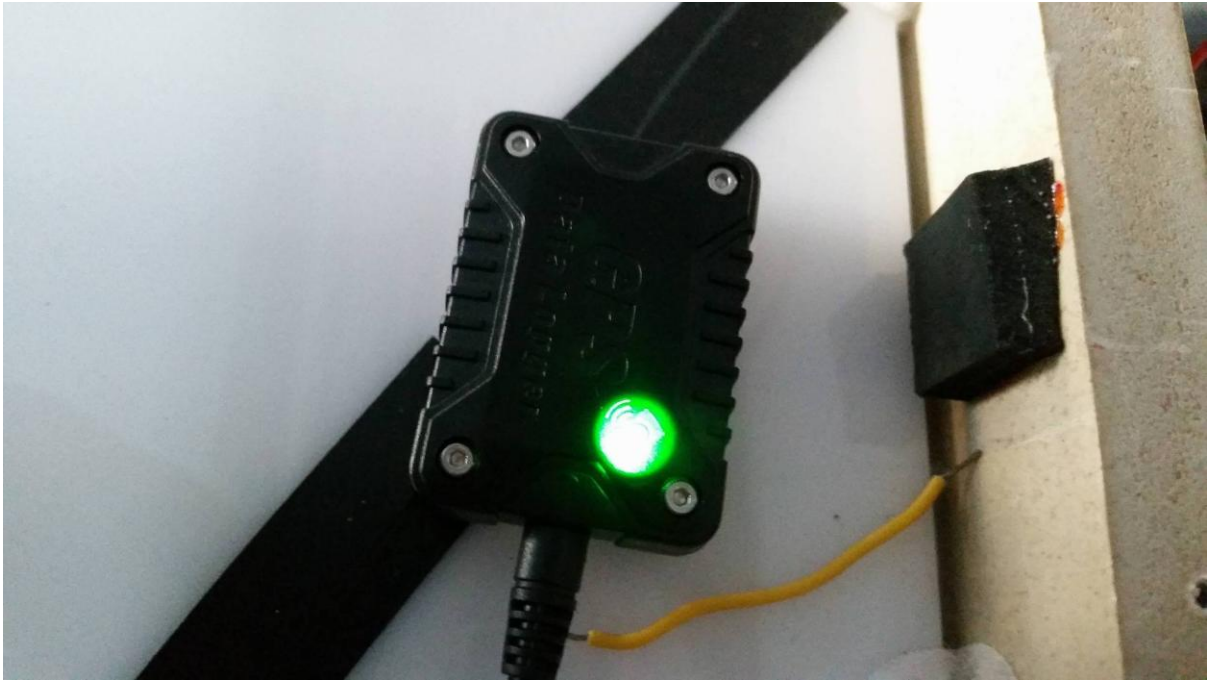


Figure 2: The GPS velocity vs time, output through the EKF, and the Integrated Acceleration.

The figure 2 above shows a large divergence in the acceleration if the GPS acceleration is obtained just through raw integration. This is why odometry was investigated in order to prevent a large divergence from appearing near immediately.



**Figure 3: The V-800 USB GPS module**

Due to the fact that the current software uses the GPS for setting the starting position before a drive, the GPS was kept for localization.

### **3.3. IMU**

The IMU employed in this project is an XSens MTi-100, which provides velocity and acceleration around the x, y and z axis, as well as pitch, roll and yaw [19]. The IMU was calibrated in previous years to fit the SAE's specific dimensions and to protect it from interference from metal. It can output values in ranges from 100Hz to 10Hz, in which 10Hz was chosen in order to provide stable enough data for the sensor fusion, and to allow the GPS module to keep pace.





**Figure 4: Xsens IMU MTi 1000**

### **3.4. Cameras**

As the REV project plans for the use of computer vision algorithms for object detection and planning, localization and vehicular odometry, two cameras have been mounted on the top of the SAE vehicle on either side of the IMU. These are for cone detection in emulation of [4], and possible other road feature detection. This is to be implemented into the system and give object coordinates, for which they need good localization in order to do this.

### **3.5. Lidar**

The Light Detection and Ranging device used is an IBEO LUX. This allows four layers of object detection for distances up to 200m, and as many as 65 objects to be tracked [20]. This communicates with other system through Ethernet cable, so TCP/IP is what is needed to perform communications with the device. Similar to the camera setup, this system of object detection and mapping also depends on the localization from the sensors being highly accurate.

### **3.6. Odometry**

Taking into account the problems with the initial odometry that were outlined in [7], the Odometry circuit was modified. The Front Wheels were achieved through using Hall Effect sensors. The Hall Effect sensors worked through the use of placing magnets on the front wheels of the car, and using the output of the Hall Effect sensors when they are closest to the magnet for their respective wheel, to become the input for a simple circuit to pickup when the Hall Effect sensors arrive in close proximity with the magnets on the wheels.

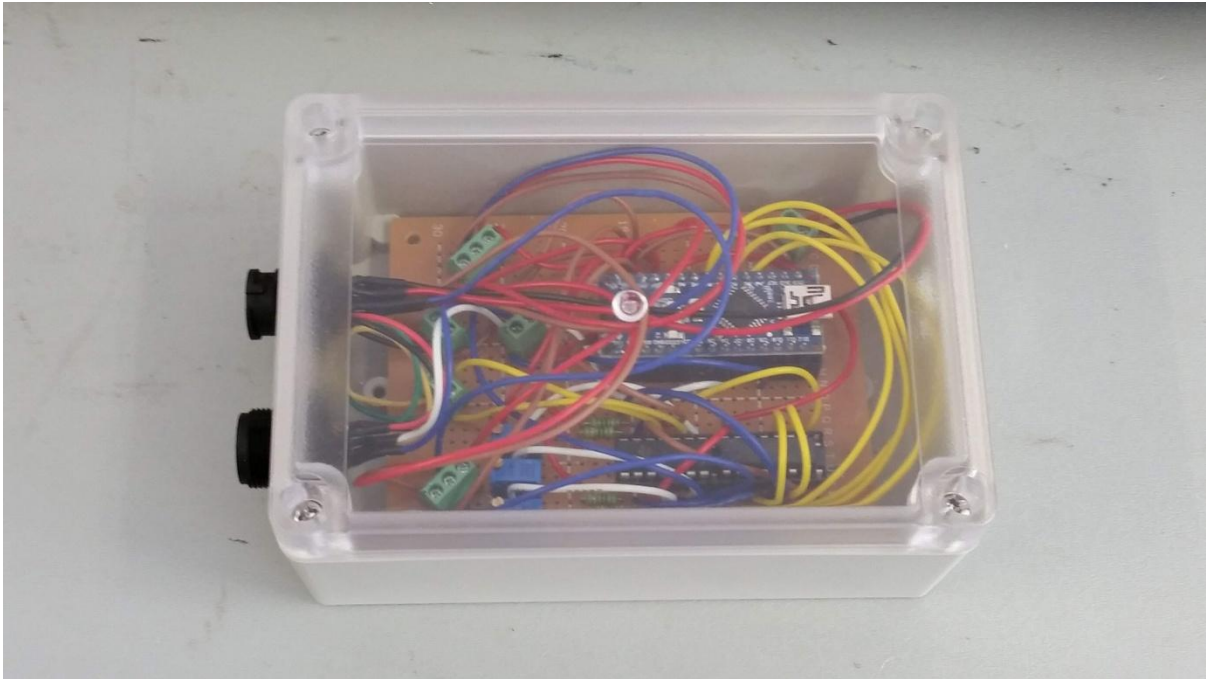


Figure 5: The Odometry Circuit.

For the rear wheels, odometry is available via the motor controllers and motor of the SAE vehicle. The motor has built-in Hall Effect sensors. The Hall Effect sensors of the rear end have to be isolated from the low level and odometry circuits in order to avoid electrical hazards. To achieve this, an optocoupler isolation circuit was utilized, that obtains the signal of the pulses of the rear wheel motor, and passes it through the rest of the Odometry circuit as a pulse train similar to the front wheel Hall Effect sensors. There was an existing optocoupler circuit, but that proved to be non viable, as there was never any output in response to input, so the circuit had to be redesigned. Later investigation found that it was due to the optocouplers being used in that isolation circuit were TRIAC optocouplers, the wrong type required for digital signals. The pulses from the hall effect sensors and the optocoupler isolation circuit goes through a comparator chip, and the outputs of the comparator chip go into both the OR gate and the analog inputs of the Arduino Nano. The Arduino Nano runs an ISR when the output of the OR gate sends a high signal to one of the pins on the Arduino Nano. The Odometry microcontroller then uses the time difference to get these interrupt signals:

$$RPM = \frac{60000}{(t_2 - t_1) * Magnets\ per\ revolution}$$

From then on a simple equation to transform the RPM of each wheel into meters per second was used.

$$WheelSpeed = \frac{2\pi r RPM}{60}$$

In this r is the radius of the wheels, of which the SAE has all four wheels at the same radius at 0.2575m. The Arduino Nano then sends the wheel speeds over to the Arduino Uno through a serial interface emulated by the Arduino Software Serial library, which can then send the wheel speeds to the NVidia Jetson TX1 board. The rate chosen for this was around 5Hz, to be a similar to the GPS and to the IMU for the fusion, and to not interrupt the other functions of the Arduino Uno.

An issue described in [7] was that the samples may be not frequent enough for the microcontroller to report accurate speeds. This was due to the Arduino Uno having several other functions that use ISRs in order to function effectively.

### 3.7. Ackerman Drive Model

After obtaining the wheel speeds from the odometry circuit, these wheel speeds need to be translated into something more useful than a magnitude of speed to be useful for the SAE vehicle. For this, a steering model is required. As the SAE vehicle utilizes Ackerman steering, using this model to get from wheel speed magnitudes and steering angle to a linear x and y velocities is needed. From [17], we can obtain this from:

$$\frac{\psi}{V} \approx \frac{1}{R} = \frac{\delta}{L}$$

Where V is the average of all four wheel speeds,  $\psi$  is the yaw angle,  $\delta$  the steering angle, R the radius of curvature, and L the distance between the front and back wheels. From the SAE vehicle, the measured L is 1.81m,  $\delta$  ranges from  $\pm 30^\circ$ , and since the SAE uses a steering linkage arrangement that keeps the left and right steering angles approximately equal, even though in reality there will be a small difference, in low speeds this will not be significant, and our steering sensor does not account for the steering angles of two wheels. After obtaining the radius of curvature, the  $\psi$ , the steering yaw can be obtained, then the slip angle,  $\beta$  from:

$$\beta = \tan^{-1}\left(\frac{l_f \tan \delta_r + l_r \tan \delta_f}{l_f + l_r}\right)$$

In this equation,  $l_f$  and  $l_r$  are the distances from the centre of gravity. For the SAE vehicle, these values are approximately  $l_f = 1.15\text{m}$  and  $l_r = 0.66\text{m}$  respectively. After obtaining the slip angle and the driving angle, the linear x and y velocities can now be calculated as:

$$x = v \cos(\psi + \beta)$$

$$y = v \sin(\psi + \beta)$$

From there on these values are stored into a vector for the sensor fusion to use. As the wheel speeds are sent from the Arduino Uno to the Jetson TX1, there needed to be programming done to capture the output and store it in order for it to be utilized around the rest of the software. The class that initially interacted with the Arduino Uno was called LowLevelSerialOutput, which utilized the boost software library to perform asynchronous serial communications with it. Changes to the code of this class were to capture the output and process it into a vehicle vector.

### 3.8 Low Level Improvements

As the automated steering, braking and acceleration system on the SAE vehicle was controlled by the Arduino UNO, changes to the programming needed to be made to make it as efficient as possible. Initially the steering control unit was running on a simple while loop which would may have could potentially cause issues when the system needs to perform a turn and it was not in that part of the code for a particular frame of time. This was moved into a hardware timer interrupt algorithm to ensure that the steering would tune itself every 150ms.

The steering PID loop had a problem in that when turning too hard to either the left or the right directions, the whole system would turn off. While the Raspberry Pi was still on as the main processor, an attempt to mitigate this was by using a large capacitor at the power source of the

Raspberry Pi, which although somewhat worked, still did not completely alleviate the original problem. Eventually, software changes to the Arduino Uno were made to prevent this. Effectively the steering range of the maximum left and maximum right angles were restricted in order to stop the steering motor controller from hitting the end stops of either side of the steering shaft. This was a success in preventing the automatic steering from turning off

### 3.8.1 Formatting

The output values of the Arduino Uno that are required to be outputted into the NVidia Jetson TX1 board outputted in a paragraph describing all the values, with many spaces and periods as separators. Each time the data is outputted, can potentially amount to strings of lengths over 200 characters. This proves to be bad, especially as the LowLevelSerialOut class processes the serial output by separating them with newline characters, which can cause packets of information that need to be displayed separately to possibly together, and that the large paragraph uses up more time and processing power in the rest of the system. The serial output from the Arduino Uno was therefore change to reflect LowLevelSerialOut, which means the values were separated via newline characters and the value names were abbreviated to save string space. This was simply done by abbreviating values that are outputted to the user. The result was being a string that was reduced in size by more than half, which can be viewed in the Appendix (low level changes).

### 3.9.1 Kalman Filter

From [5], the Kalman Filter was accomplished by importing a 3<sup>rd</sup> party library called OpenCV. This library includes software for matrix calculations, and has its own implementation of a Kalman filter class. The code of the Kalman Filter allows it to be modified to work as an EKF [21]. This was done to provide

### 3.9.2 Extended Kalman Filter

The application of extended Kalman filters is common in the field of localization. The EKF is an extension to the normal Kalman filter, which attempts to compensate for the Kalman Filter's weakness in estimation for non-linear systems. A system is linear if it follows the rule that:

$$af(x) = f(ax)$$

As the SAE vehicle will be travelling in angular and linear directions, it follows that the system can be assumed to be non-linear. The Kalman filter is optimal given that the system it is being used on is linear. Otherwise alterations of the Kalman Filter will need to be utilized.

As with the Kalman filter, the EKF requires a model to be chosen for it to work optimally. When compared with the regular Kalman Filter, the EKF relies on the Jacobians of the state transition function and the observation function to accurately model it. As this filter is being applied to find the correct position, velocity and accelerations of the x and y coordinates, the state transition matrix was in [5] was defined as:

$$F = \begin{matrix} 1 & dt & \frac{dt^2}{2} \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{matrix}$$

The EKF defines  $F_{k-1}$  as:

$$F_{k-1} = \left. \frac{\partial f}{\partial x} \right|_{x_{k-1}|k-1, u_{k-1}}$$

So for the EKF, using the state transition matrix for linear position, velocity and acceleration, this comes to be:

$$F_{k-1} = \begin{bmatrix} 0 & 1 & dt \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

The observation matrix is also been defined as the Jacobian of the observation function, which for this paper will be assumed as:

$$H_k = \left. \frac{\partial h}{\partial x} \right|_{x_{k-1}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This observation matrix is an Identity matrix in our case due to the face that the EKF will use one sensor per variable measurement.

### 3.9.3 Sensor Fusion

Sensor fusion is the most significant task in localization, as it is the test of the effectiveness of the sensors, as well as the effectiveness of the filtering algorithms employed. The previous sensor fusion algorithm involved GPS and IMU as the primary sensors [13],

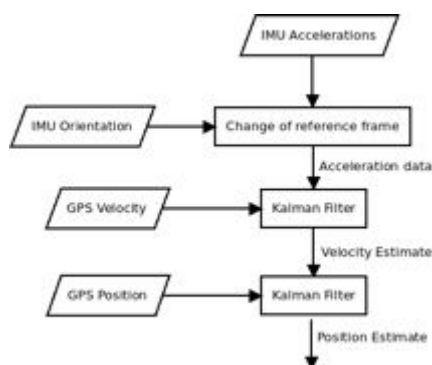


Figure 6: Original Sensor Fusion Algorithm [5].

As outlined in figure [5], the sensor fusion algorithm starts off with obtaining the IMU orientation and acceleration frame to change the reference frame, then the acceleration and GPS velocity are processed into a Kalman filter, then the output of that is put into another Kalman filter with the GPS position measurements. Using the Columbus V-800 USB GPS has shown to be unreliable, as it cannot get as many locks on satellites as the Piksi RTK-GPS can, and connections to satellites are easily interrupted by walls.

With the addition of odometry, the Kalman Filter now has a more stable input, over the GPS which depends on outside systems such as satellites. An alteration to the Kalman Filter is done by using the velocity vector transformation via the Ackerman steering method from the odometry and using that as the input into both the velocity-acceleration and position-velocity Kalman Filters again. The system is described below:

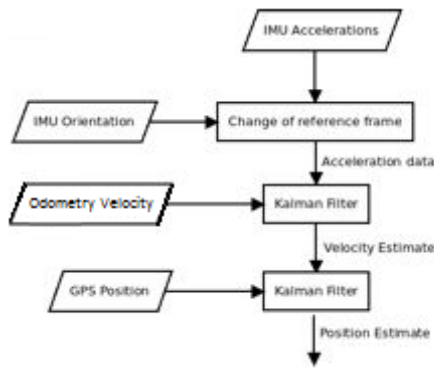


Figure 7: Modified Sensor Fusion Algorithm, with Odometry as the velocity measurement.

From then on after this sensor fusion algorithm has been verified, the next step can be taken by directly implementing the EKF. The control chart below describes this.

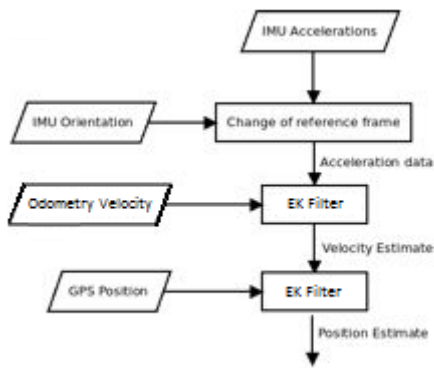


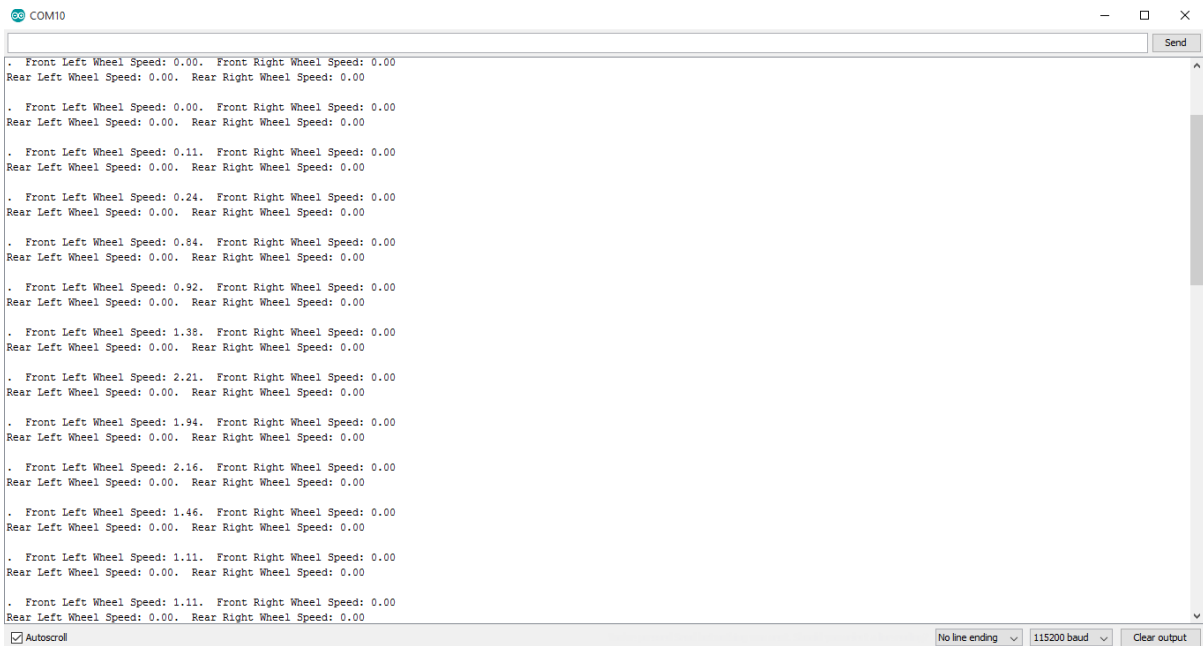
Figure 8: Modified Sensor Fusion Algorithm with Odometry and EKFs.

After the Kalman filter changes to include the SAE Vehicle odometry measurements were made, the class BoeingKalman had to be changed in order to emulate an EKF, according to [21]. Using these sensors, the system can then use dead reckoning to record the current position, after being filtered by the Extended Kalman Filter.

## 4. Results

### 4.1. Odometry

The Odometry Circuit that was made was successful in extracting the wheel speed of each wheel. The new optocoupler isolation circuit between the motors, motor controllers and the odometry circuit succeeded in recording the velocity measurements. The Arduino Nano and Uno were able to communicate successfully via software serial without any issues to the safety actions the Uno has to perform, as well as the steering, braking and the acceleration of the SAE vehicle.

The image shows a screenshot of an Arduino serial monitor window titled 'COM10'. The window contains a list of data points, each representing a set of wheel speeds. Each line of data is formatted as follows: '. Front Left Wheel Speed: [value]. Front Right Wheel Speed: [value] Rear Left Wheel Speed: [value]. Rear Right Wheel Speed: [value]'. The values for the front wheels vary, while the rear wheels are consistently 0.00. The data points are: 1. Front Left: 0.00, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 2. Front Left: 0.00, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 3. Front Left: 0.11, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 4. Front Left: 0.24, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 5. Front Left: 0.84, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 6. Front Left: 0.92, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 7. Front Left: 1.38, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 8. Front Left: 2.21, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 9. Front Left: 1.94, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 10. Front Left: 2.16, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 11. Front Left: 1.46, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 12. Front Left: 1.11, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. 13. Front Left: 1.11, Front Right: 0.00, Rear Left: 0.00, Rear Right: 0.00. At the bottom of the window, there are controls for 'Autoscroll' (checked), 'No line ending', '115200 baud', and 'Clear output'.

**Figure 9: Wheel Speed output of Odometry circuit via Arduino serial monitor.**

The interrupt software model proved to have issues. This is mainly due to the logic of the controller. As the isolation circuit inverter digital highs and lows of the rear wheel hall sensors, this would cause a constant interrupt if the isolation circuit was not connected and working. Changing the software model into a polling algorithm improved the reliability of the odometry, as the logic issue was superseded by not depending on an interrupt.

An issue that arose from [7] was that the ISRs cause by the Odometry logic circuit had a very inconsistent pulse count. How much of this was due to the faulty wiring of the Low Level Control circuit is unknown, but it remains a valid concern, though due to the Odometry counter circuit not being required to perform functions other than besides counting and one way serial communications, the 16MHz microprocessor of the Arduino Nano should be more than adequate enough to perform the pulse counting for the car.

The LowLevelSerialOut was able to obtain the wheel speed measurements from the Arduino Uno serial interface. The results of the translation of the wheel speeds into a velocity vector showed results that were in line with the wheel speeds. The problem with the wheel speeds is that they are not in close range with each other, which is probably due to calibration errors from the odometry circuit. This will have to be corrected in the future. The rear wheel speeds are likely the more correct ones, as the front wheel sensors are affected by the Hall Effect sensors being non-uniformly close to their respective wheel's magnets. Despite this, the wheel speed sensors have been partially integrated into the software. The odometry circuit may require more adjusting before it becomes a really accurate reader for the speed of the SAE vehicle.

|  |                         |                                  |                                       |   |                        |
|--|-------------------------|----------------------------------|---------------------------------------|---|------------------------|
| Man Sts<br>0                             | LowLevelSerial Sts<br>1 | GPS Speed<br>0                   | IMU Heading<br>328.37957169433923     | Fused Speed<br>0                        | Odo VelY<br>0.98310101 |
| Auto On<br>0                             | Safety Mode<br>0        | GPS Track Angle<br>0             | Desired Bearing<br>0                  | Fused Heading<br>3.9525251667299724e-32 | AdvancedPath Actv<br>0 |
| Auto Run<br>0                            | Steer Posn<br>0         | GPS Time<br>0                    | NextWaypoint<br>0                     | Wheel Speed FL<br>0.349999994           | HBSerial RSSI<br>0     |
| Trip Sts<br>0                            | Throttle/Brake Lvl<br>0 | Datum Lat<br>-31.980568999999999 | Fused X Pos<br>4.6590914952325739e-31 | Wheel Speed FR<br>0.349999994           |                        |
| HB Sts<br>0                              | GPS State<br>1          | Datum Long<br>115.817807         | Fused Y Pos<br>9.4860604001519336e-32 | Wheel Speed BL<br>1.04999995            |                        |
| CarNet Sts<br>Waiting to accept a connec | GPS Lat<br>0            | Offset Lat<br>0                  | Fused X Vel<br>2.7099416455962383e-31 | Wheel Speed BR<br>0.860000014           |                        |
| SafetySerial Sts<br>1                    | GPS Long<br>0           | Offset Long<br>0                 | Fused Y Vel<br>7.1303523258804442e-67 | Odo VelX<br>-1.70139289                 |                        |

Figure 10: Wheel Speeds and Odometry X and Y velocities from the SAE web interface.

## 4.2. Low Level System Changes

The changes made to the Low Level Arduino Uno provided several benefits. Though at first, the PID loop constants that the steering used,  $K_p$ ,  $K_D$  and  $K_i$  needed to be changed to reflect the new system, as the previous values had were causing significant overshoot and increased the settling time significantly. The changes to the output formatting allowed for more robust output and better reflected the embedded system of it.

The change in the low level control software the involved limiting the range of the steering proved to stop the steering motor controller from drawing too much current and turning the system off abruptly. This also will preserve the SAE vehicle's steering shaft, as the steering motor controller will not be able to hit the stoppers at the ends of the steering shaft. A small downside to this is that the turning circle of the SAE vehicle will be reduced,

## 4.3. Kalman Filters

The Kalman Filters with and without odometry were implemented in software, but the tests did not really help with the localization, indicating that the fusion class has some issues in software. Each of the Kalman Filters variants (GPS-IMU and GPS-Odometry-IMU) were implemented in the code and are able to be compiled, indicating that the issue is within the fusion class itself. This will need to be fixed as soon as possible for the SAE vehicle to be able to localize itself.

## 4.4. EKF

The EKF with Odometry model has not being fully implemented in the software, though the BoeingKalman class was adjusted to be an EKF filter. As the regular fuser class ran into issues, the EKF was not able to be implemented in software.



## 5. Conclusion

The beginning of the project determined that there were some much needed improvements in the localization of the SAE vehicle. The quality of the localization was initially very good, due to the use of an RTK-GOS module, and a working sensor fusion class, to the degree that the SAE vehicle was able to drive via GPS waypoints.

The initial odometry circuit and software that was constructed in [7] was so constrained that it could not be a reliable measurement apparatus. The new odometry hardware and software is significantly less restrained and more reliable. Through some minor calibrations of the magnets and potentiometers, the odometry circuit can become a very reliable measurement device for the SAE vehicle's localization.

The EKF has been partially implemented in software through altering the BoeingKalman class. It has not been fully implemented in the fusion class, which will need to be done in the future. If the EKF and sensor fusion can be implemented correctly, there is a significant promise for the localization capabilities of the SAE vehicle.

## 6. Future Work

There is a significant scope of future work that can be to the localization system of the SAE vehicle to improve the autonomous driving capabilities of it. The first would be being able to compare the sensor fusion variants to find the most optimal use, although as the per paper [9] shows, as long as the non linear system linearization done by the EKF does not diverge too much, that EKF and Unscented Kalman Filters perform similarly. Though as driving is a largely non linear environment, implementing an Unscented Kalman Filter would probably provide better measurements, and it more simple to implement from [8].

The Odometry can be improved by changing the model currently used to calculate the x and y velocities into something that reflects reality a lot more. A more complex model is described [22], where the individual velocities of the x and y axis of the wheels and the angular velocities are taken into account, as well differing models for high and low speeds. The software code could also be potentially changed so that the Odometry completely replaces the need for a GPS, by providing acceleration and position measurements. Though potential changes in the low level odometry code would have to be done in order to refine the odometry for this use, which would include a better turning model for the estimate of the slip angle. Providing a better estimate for the velocity from the odometry system will make it more suited to be able to perform the function that it was meant to do; allow the SAE vehicle to perform drives for periods at a time without a functioning GPS system.

As the Piksi GPS system from Swift Navigation was replaced with the Columbus V-800 GPS module, a cheaper, less precise option that does not use RTK-GPS, the GPS navigation has suffered as a result of this. This was mainly due to the Piksi being newer hardware, which had less documentation available for it, as well as a USB reliability issue. With the release of firmware version 1.2 [23], the USB issue seems to have been remedied, and the Piksi can be potentially moved back onto the SAE vehicle. The previous software for it may have to be modified if this were to happen.

The now stable odometry can now provide a several advance driver assistance systems. These include anti-lock braking, rollover protection and electronic stability control, which would also improve the safety of the car for future students that work on it. These advanced driver assistance systems would probably requires a microcontroller with a greater processing power, meaning the Arduino Uno would have to be changed out. From [24], the list of Arduino products similar to the Uno but have higher processing power are the Zero, Due, Yun and MKR1000 among the variants of Arduino products available. Any variant of these chosen would result in some reworking of the Low Level Control

circuit needing to be implemented. The most powerful of these variants is the Arduino Yun, which has a 200MHz microcontroller, 64MB of RAM, an Ethernet port, which would allow it to perform probably all advanced driver assistance system functions if needed to.

Currently the software architecture is being revised in order to construct a make each class less interlinked between each, a problem that occurred with the implementation in [5]. This is make the code more accessible and easier to read.

To improve the reliability of the circuits, removing the prototyping boards out of the SAE Vehicle in place for properly made PCBs would improve the circuits considerably. This is especially true for the Low Level Control circuit, and the Safety Supervisor circuit, but can be extended to all of the low level circuits.

With the addition of the NVidia Jetson Titan running as the SAE vehicle's main control system, the use of SLAM related techniques becomes a real possible to test and perform without needing to worry about processing power too much. This processing, with the cameras set up on the SAE, allows the implementation of Visual Odometry to be added as a feature, to help with both the localization, as well as the object detection.

The final improvements to localization that could potentially be made to the SAE Vehicle is through the addition of a particle filter based system. A recent paper by Chen et. al shows how much less the error is in a particle filter than in a EKF system. If this could be implemented correctly, the SAE Vehicle would probably not need any other improvements in its localization.

## 7. References

- [1] "“Phantom Auto” to Be Operated Here.” *The Free-Lance Star*. *Google News Archive*. 17 June 1932. [Online]. Available: <https://news.google.com/newspapers?id=PthNAAAAIIBAJ&sjid=yYoDAAAAIIBAJ&pg=6442,38790> 17. [ Accessed: 26 May 2013.]
- [2]R. Rosen, "Google's Self-Driving Cars: 300,000 Miles Logged, Not a Single Accident Under Computer Control", *The Atlantic*, 2012. [Online]. Available: <https://www.theatlantic.com/technology/archive/2012/08/googles-self-driving-cars-300-000-miles-logged-not-a-single-accident-under-computer-control/260926/>. [Accessed: 17 May 2017].
- [3] Caddy, Becca. "Toyota To Launch First Driverless Car In 2020". WIRED UK. N.p., 2015. Web. 25 May 2017.
- [4] "Formula Student Germany: Autonomous Driving at Formula Student Germany 2017", *Formulastudent.de*, 2017. [Online]. Available: <https://www.formulastudent.de/pr/news/details/article/autonomous-driving-at-formula-student-germany-2017/>. [Accessed: 20- Oct- 2017].
- [5] T. Drage, "Development of a Navigation Control System for an Autonomous Formula SAE–Electric Race Car", Final Year BE Project Thesis, UWA, 2013
- [6] T. Drage, J. Kalinowski and T. Bräunl, "Integration of Drive–by–Wire with Navigation Control for a Driverless Electric Race Car", REV Project thesis, UWA, 2013
- [7] C. Blignaut, "Methods for Improving the Absolute Localisation of an Autonomous SAE Vehicle," REV Project Thesis, UWA, 2015

- [8]S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation", Proceedings of the IEEE, vol. 92, no. 3, pp. 401-422, 2004.
- [10] S. Sun and Z. Deng, "Multi-sensor optimal information fusion Kalman filter", *Automatica*, vol. 40, no. 6, pp. 1017-1023, 2004.
- [9] Chandrasekar, J., Ridley, A. and Bernstein, D. (2007). A Comparison of the Extended and Unscented Kalman Filters for Discrete-Time Systems with Nondifferentiable Dynamics. 2007 American Control Conference.
- [11]E. Al Khatib, M. Jaradat, M. Abdel-Hafez and M. Roigari, "Multiple sensor fusion for mobile robot localization and navigation using the Extended Kalman Filter", *2015 10th International Symposium on Mechatronics and its Applications (ISMA)*, 2015.
- [12] Barbosa, D., Lopes, A. and Araujo, R. (2016). Sensor fusion algorithm based on Extended Kalman Filter for estimation of ground vehicle dynamics. *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. [online]. [Accessed 15 Aug. 2017].
- [13] Jinglin Shen, D. Tick and N. Gans, "Localization through fusion of discrete and continuous epipolar geometry with wheel and IMU odometry", Proceedings of the 2011 American Control Conference, 2011.
- [14]"NVIDIA Jetson Modules and Developer Kits for Embedded Systems Development", Nvidia.com, 2017. [Online]. Available: <https://www.nvidia.com/object/embedded-systems-dev-kits-modules.html>. [Accessed: 14- Oct- 2017].
- [15]100W Single Output DC-DC Converter. Mean Well, 2017.
- [16]"Raspberry Pi 3 Model B - Raspberry Pi", *Raspberry Pi*, 2017. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. [Accessed: 16- Sep- 2017].
- [17] R. Rajamani. Vehicle Dynamics and Control. New York: Springer-Verlag, 2006, ch. 2.
- [18]*User's Manual Specially for Columbus V-800+ GPS data logger*. Columbus GPS, 2012.
- [19] Xsens Technologies B.V. (2010, Oct.). "MTi and MTx User Manual and Technical Documentation" [Online]. Available: [http://www.xsens.com/images/stories/products/manual\\_download/Mti\\_and\\_Mtx\\_User\\_Manual\\_and\\_Technical\\_Documentation.pdf](http://www.xsens.com/images/stories/products/manual_download/Mti_and_Mtx_User_Manual_and_Technical_Documentation.pdf). [Accessed: Mar. 6, 2013]
- [20] ibeo Automotive Systems GmbH (2013, Sep.). "IBEO Lux Datasheet" [Online]. Available: [http://www.ibeo-as.com/ibeo\\_lux.html](http://www.ibeo-as.com/ibeo_lux.html).
- [21] "Motion Analysis and Object Tracking — OpenCV 2.4.13.4 documentation", Docs.opencv.org, 2017. [Online]. Available: [https://docs.opencv.org/2.4/modules/video/doc/motion\\_analysis\\_and\\_object\\_tracking.html?highlight=kalman%20filter#KalmanFilter](https://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html?highlight=kalman%20filter#KalmanFilter). [Accessed: 14- Oct- 2017].
- [22]R. Longoria, "Turning Kinematically", The University of Texas at Austin, 2015.

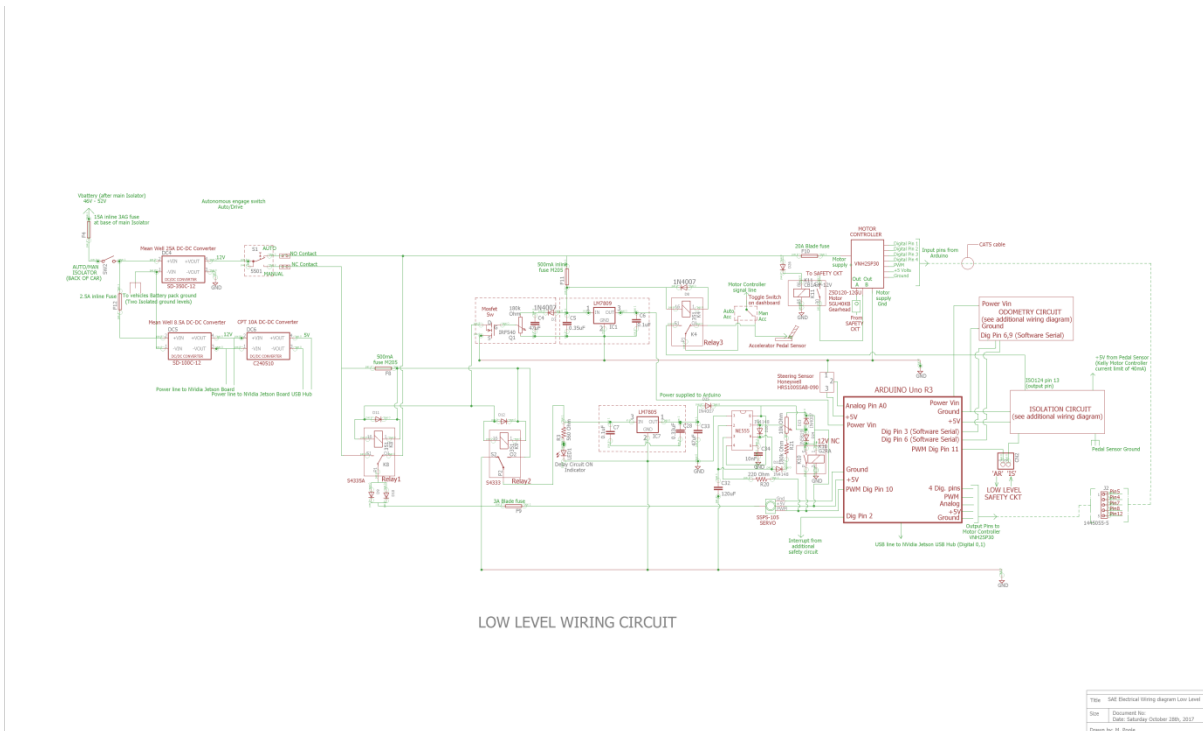
[23]PIKSI ® MULTI FIRMWARE 1.2 RELEASE. Swift Navigation, 2017.

[24]"Arduino - Compare", *Arduino.cc*, 2017. [Online]. Available: <https://www.arduino.cc/en/Products/Compare>. [Accessed: 13- Oct- 2017].

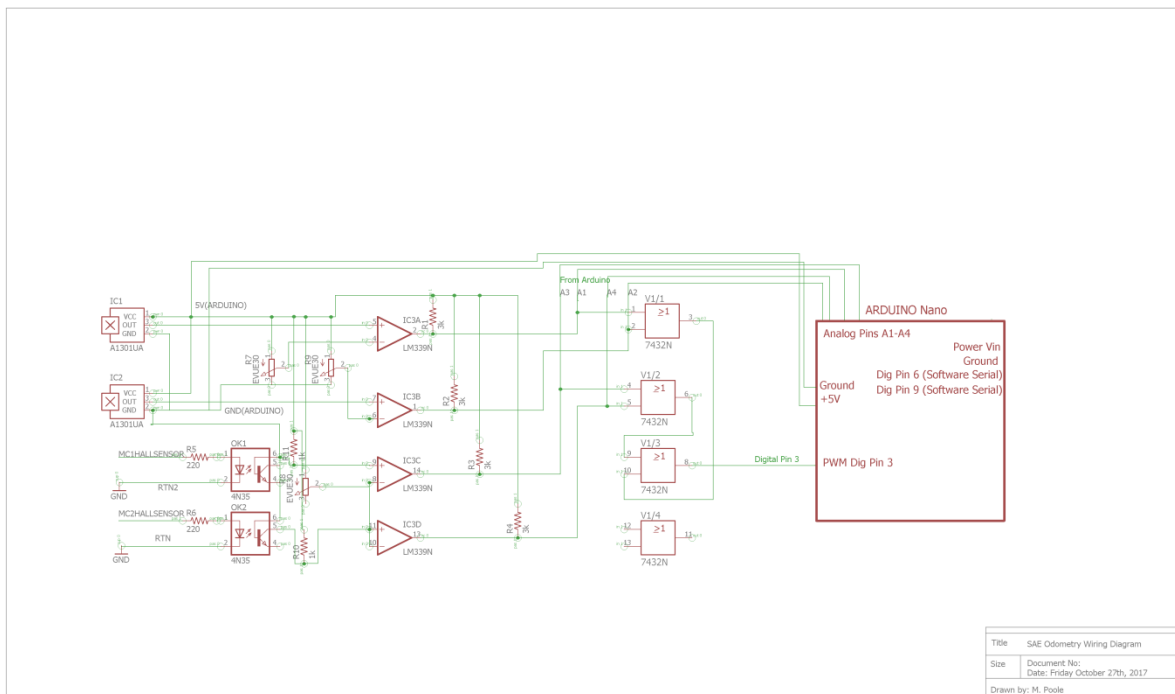
[25] X. Chen, S. Song and J. Xing, "A ToA/IMU indoor positioning system by extended Kalman filter, particle filter and MAP algorithms", *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2016.

## **8. Appendices**

### **Appendix A - Low level circuit changes**



## Appendix B – Odometry and Opto Coupler Isolation circuit



## Appendix C – Low Level Code Changes

1. /\* Copyright 2013 Thomas Drage