Robin Rai

Registration number 100242165

2022

Electronically Governed Mountain Bike Suspension

Supervised by Edwin Ren

University of East Anglia
Faculty of Science
School of Computing Sciences

Abstract

Modern mountain bikes feature suspension capable of tackling rough terrain with ease. However, this suspension can negatively affect pedalling efficiency over smooth terrain. This can negatively affect times in racing, or reduce the distance a rider can ride for leisure. Ideally, suspension should be disabled, or "locked out" when traversing smooth terrain, and active while traversing rough terrain, to both maximise traction and stability over rough terrain and maximise pedalling efficiency over smooth terrain. Typically until recently this has been done manually, with the use of manual lockout controls present on suspension components or the handlebars. Commercial electronic systems made to automate this have recently been launched to quiet reception, due to their high cost and requiring proprietary suspension components to function. This project aims to solve this suspension problem with an embedded system that is affordable and widely compatible with a range of suspension components, and goes through the process of creating a basic but expandable system that devises methods of measuring and quantising different types of terrain and rider input, and enables or disables the bike's suspension itself accordingly. A functioning prototype was successfully built which reaches the goals of the project. It is expandable, inexpensive, and widely compatible. A final system is orders of magnitude less expensive than commercial offerings, with a total cost of less than £100.

Acknowledgements

A big thanks to my family, friends, and university faculty members.

Contents

List of Figures

List of Tables

1. Introduction

Featuring front and often rear suspension, mountain bikes (MTBs) are bicycles designed for riding and tackling rough terrain. The suspension allows the wheels to move independently from the bicycle over bumps, and absorb shocks. While this increases stability, traction and comfort over rough terrain, it is detrimental on smoother terrain as some energy from pedalling can be wasted spent compressing the suspension instead. The most widely accepted solution in the industry takes the form of suspension lockout. This is a feature found on suspension components that allows said suspension to be disabled or "locked out", via a control found on or near the suspension, for traversing smooth terrain. These levers can pose some issues. To operate them efficiently, the rider must take a hand off of the handlebars and reach down to use them, which can be both inconvenient and potentially dangerous, especially when terrain changes frequently. While solutions such as handlebar mounted controls have been available, this project aims to automate this task in real-time via an embedded system, that will gauge rider intention and the terrain being traversed to govern and control the lockout levers accordingly itself, allowing riders to concentrate on riding. Recently electronically controlled and governed suspension have been unveiled and made commercially available by the two biggest mountain bike suspension manufacturers, their aim to make the riding experience as efficient and enjoyable as possible. Attempts to do this in the past were aimed purely at racing and have commercially failed, due to limitations, lack of adoption and cost. All of these systems share a large flaw - they are all very expensive and proprietary to a particular suspension system or series of MTBs. The commercial motivation for my system is to accomplish the same pedalling efficiency benefits of these systems, but in an inexpensive way, whilst being widely compatible to retro fit to a large range of pre-existing suspension systems and bicycles. Another motivation is to build a system ripe for expanding on, adding more features or methods of function without increasing cost or complexity drastically to the user, for future expansion.

The final system is a functional embedded system that uses an accelerometer to gauge terrain smoothness, and a hall effect sensor to gauge pedalling input. The front and rear suspension is then controlled physically with their respective servos. A single-board computer is used to determine whether or not the suspension should be active or locked in real-time, and a mode button and mode status LED allows for the rider or

user to manually override the system with their choice. Measuring terrain inclination or steepness was unfortunately cut due to an oversight late into development. The system can be easily transferred to more suitable hardware for production and is an excellent platform for future expansion. The system costs less than £100, a fraction of the cost of the available commercial options, and can be adapted to a wide range of suspension systems and brands.

2. Background and Related Works

2.1. Sensors and Components

The system requires input and from the real-world and some form of output. For measuring bumps or classifying the terrain an accelerometer will be used, as a bump can be simply read as a sudden and temporary change in acceleration. For measuring inclination, a value can be derived from triangulating the direction of gravity from the three axis an accelerometer measures as shown in Fig. 1, or a gyroscope can be used to measure angular change from a starting point.

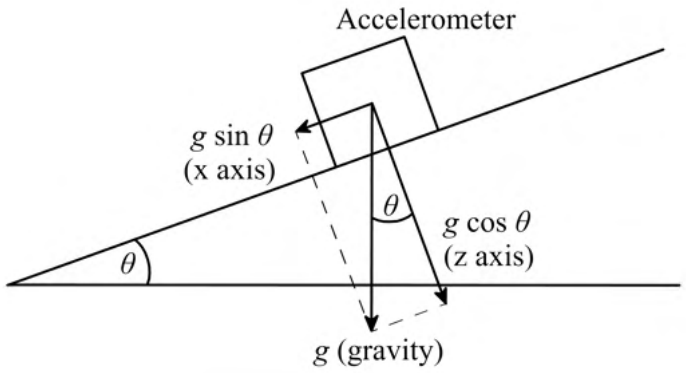


Figure 1: Calculating the angle of inclination (θ) from the X and Z axes an accelerometer would measure.

To gauge rider input a hall effect sensor can be used to detect whether or not the rider is pedalling, as if they are not pedalling improving pedalling efficiency is obviously not required.

For output, servos will be used to physically turn the lockout levers/knobs present on the suspension.

For extra accessories, a button and LED can be used for user control and to relay information back to the user. For example, the button could toggle between different modes the system could have, such as manual control and automatic control.

Any sensors used would need to be handled appropriately, as they can be susceptible to noise, interference and drift. These issues and their mitigation were researched in advance. Accelerometers for an example measure acceleration, but do not produce a clean output of bumps, instead they will be large values among small and constant noise from negligible vibration, as well as possible interference. Research on sensor filtering such as the Kalman method was performed (Welch and Bishop, 2006) (Welch and Bishop, 2001), along with where to place the sensor(s) for the best output. For example, data gathered by (Simon and Ahmadian, 2001) suggested that mounting a given accelerometer on the frame of a vehicle rather than an unsuspended area of the vehicle can potentially filter out different types of vibration, at the cost of potentially losing data on the terrain. Experimentation was later performed to see the best arrangement for this particular system, done by logging and analysing sensor output. Potentially classifying types of terrain rather than a more rudimentary binary output has also been looked into. For example, one can train Hidden Markov Models (HMMs) on sample data and use it to classify input (Wang et al., 2014) (Wolf et al., 2005). Research on the other sensors such as gyroscopes showed they can suffer from drifting from their original measurements (Boonstra et al., 2006) (Luinge et al., 1999), and that hall effect sensors that can be used for measuring pedalling activity can have either binary or granular readings(Ramsden, 2006). Experimentation will have to be performed on the hall effect sensor to get an accurate cadence reading. Research was also performed on actually using these components on a technical level. Research into servos used for physically controlling the suspension has also been performed, as they are controlled not by voltage, but by a pulse. Powering the servos and the rest of the components has also been addressed, as the controller board could potentially not provide enough power.

Research was performed on what microcontroller was to be used for the project. a Raspberry Pi proved to be a good platform that features a microprocessor in addition to the microcontroller, to allow for possible expansion of the system with more complicated features such as bump classification and learning. It also allows for programming the board entirely on it, requiring no compiling or board flashing, streamlining the

project's development.

2.2. Existing Works

Setting a baseline on how the system should function from a user and technical perspective requires analysing previous and existing works, including commercial and bespoke designs. Strengths and weaknesses of each system were considered, changing and improving my own design to fit untapped areas.

2.2.1. RockShox EI Shock System

The first existing and commercial system is the RockShox EI Shock system (Cunningham, 2012). Developed by both Suspension manufacturer RockShox and bike manufacturer Lapierre and released in 2012, the EI Shock was an electronically controlled rear suspension shock made for racing/pedalling focused Cross Country (XC) mountain bikes. It was available commercially as part of bike line ups from three bike manufacturers.

The system determined whether or not the rider was pedalling over smooth terrain or traversing rough terrain, and would open or lockout the rear suspension accordingly. For input data, it features two accelerometers, one being mounted on the lower legs of the front suspension (unsuspended), and one being mounted at the stem of the bike (suspended). This setup allowed for both suspended or sprung and unsprung acceleration to be measured and compared, as well as rider input to be optimally separated from bumps and ignored. It also featured a hall effect sensor at the pedals to detect pedalling. The microcontroller and battery were in a separate component mounted to the bike. The premise of this design is for the bump measurements to take place at the front of the bike, and control the rear shock before the bump arrives at the rear of the bike. Performance metrics are difficult to find for all of these systems, but the EI system's polling rate (how often the sensors are read) is 1000Hz, while the servo can be operated within 10 milliseconds. The servo takes longer to move than this however. An overview diagram can be seen in Fig. 2.

The two major weaknesses of this system are that firstly, the front suspension remains uncontrolled by the system and is therefore wasted potential. Secondly, the system was proprietary and limited to select bicycles, relegating the system to obscurity. Another negative is user hassle - while keeping a battery charged is a must for all these systems,

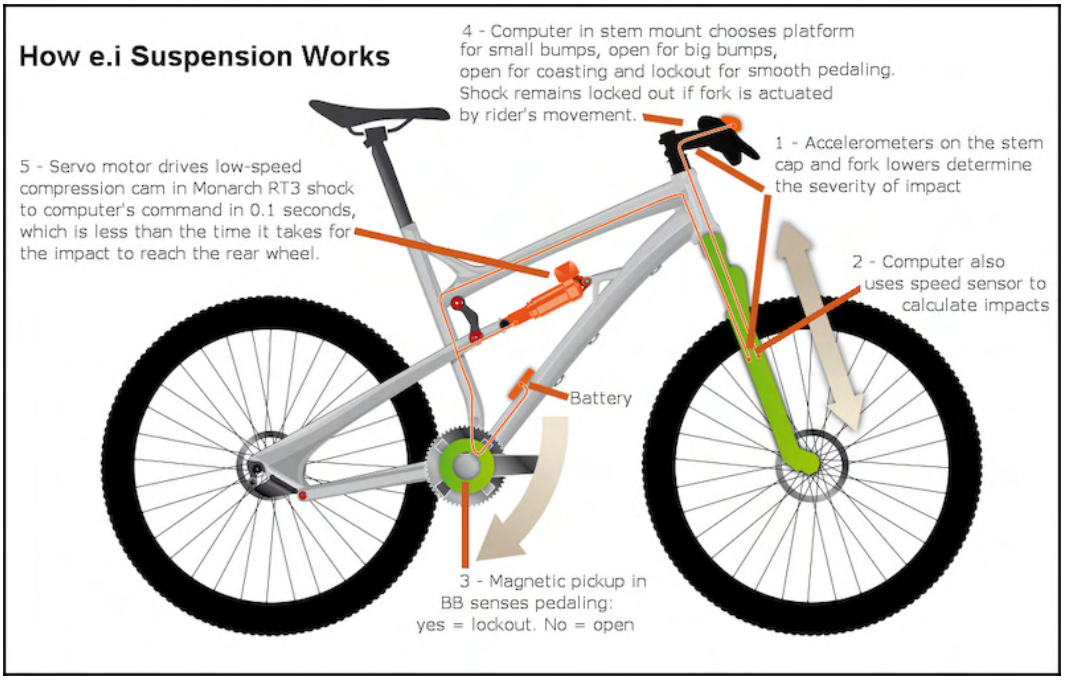


Figure 2: A system overview on the Rockshox EI Shock system (Cunningham, 2012).

there are many cables and components splayed around the bike, making user maintenance of the bike potentially difficult.

2.2.2. LiveValve

The second commercial system is far more recent. It is the LiveValve system developed by Fox Factory Suspension and released in 2019 (Cunningham, 2018). It features front and rear suspension with electronically controlled compression dampers, allowing for more granular control compared to a binary suspension lockout. For input data, it utilised a pair of accelerometers mounted front and rear of the bike, and also measures terrain inclination. It also has its battery and microcontroller fitted on the bike in a box compartment. The system can respond to terrain in 3 milliseconds.

It also features integration with smartphones and bicycle trip computers, and allows for changing preferences and biases for the suspension output. It is tightly integrated with the bicycle and requires minimal setup.

A large negative to the system is that it is once again frame dependent and proprietary, and also integrates its electronics with the suspension itself, a very costly set of

components. Prices roughly start at $3000 for a kit (Cunningham, 2018). It also shares the negative of featuring many cables and components for the user to deal with.

2.2.3. Flight Attendant

The final commercial system is the RockShox Flight Attendant system. (Kazimer, 2021) It was released in 2021, and is very similar to LiveValve, with the main difference being that the system is fully wireless, and fully integrated into the suspension - it does not feature separate microcontroller or battery components, making the system much more accessible for those who want a simple and clean riding experience. It features accelerometers in the front and rear suspension, along with a cadence sensor. It features three suspension modes for various riding scenarios - fully active, firm, and fully locked out. It has an app and button interface for rider customisation and comfort biasing. For example, the user can set a bias towards the system keeping the suspension locked out more often than the default setting. It also features a manual user controlled and temporary override mode. Unlike the LiveValve, the RockShox system can seemingly be fitted to any bike. Due to the wireless, the system polls its sensors every 5 milliseconds.

It shares some of the same negatives as the LiveValve also - it is proprietary and integrated into the suspension, though it is not limited to an offering of bikes. The system being wireless also doubles the battery count.

2.2.4. Commercial Works Analysis and Thoughts

From these three options many similarities can be seen. Firstly the basic outline of the system - similar forms of input such as accelerometers to gauge terrain, and hall effect sensors to see if the rider is pedalling. Servos are used to adjust the suspension components accordingly. They all also feature fast response times and the ability for a rider to personalise the system's functionality. This shaped the basic structure of my system - measuring terrain via accelerometers and controlling the suspension via servos was a must, and said arrangement of the sensors will have to be experimented on to see what results in the best result. Expanding the system with methods such as inclination and pedalling sensing would be ideal additions also, and a fast response time from terrain to action should be achieved. App connectivity can be considered but is likely out of scope for this project.

These commercial systems also share some negatives that should be ideally avoided.

Firstly of the slight criticism or general consensus that these systems are potentially yet another hassle to deal with for riders, along with other recent electronic bike components, requiring tasks such as charging components and fiddling with apps. As from a rider's or user's perspective the sole purpose of these systems is to improve the enjoyment and efficiency of riding, simplifying the end user experience as much as possible is ideal - after initial setup all that should be required of the user is to charge the battery and turn the system on.

2.2.5. Kolotoff's Bespoke System

Finally, the last existing system is a bespoke design by Vladimir Kolotoff, a now software engineer. His bespoke system from 2012 manipulated his bicycle's pre-existing suspension via their lockout levers. (Tyler, 2012) For input it featured an accelerometer mounted on the front fork, a speed sensor (using a hall effect sensor) and a cadence sensor. This fed into a microcontroller and battery to control two servos mounted to the front and rear suspension. It also featured buttons and a smartphone app to configure thresholds and settings. (Kolotoff, 2015) A system overview can be seen in Fig. 3.

Figure 3: A system overview of Kolotoff's system (Tyler, 2012).

The pros of this system were that it could be adapted to any preexisting suspension so long as they had lockout or damping controls, and that it could be made quite inexpensively, as microcontrollers and the other components are inexpensive today.

The cons are that for the hypothetical average user, it would be quite difficult to set up given the nature of the bespoke design. The basis for the design seems quite suitable however with its simplicity and cost, and streamlining it would produce a desirable system.

3. Design and Planning Choices

3.1. System Overview and Diagram

An initial overview diagram was made near the start of the project, shown in Fig. 4. This can aid in conveying the design and premise of the system to others more clearly. The system later on through development changed, but this initial design consisted of a microcontroller, battery, a pair of servos, a pair of accelerometers, a gyroscope, and a hall effect sensor. An accelerometer was planned to be mounted to an unsuspended and foremost area of the bike, to record the terrain as rawly and as immediately as possible. The accelerometerwas also to measure inclination or how steep the ridden terrain is, as when climbing steep inclines more efficiency would be desirable and should skew the system's output as a result. A second accelerometer was to be mounted on a suspended portion of the bike, such as the handlebars. Depending on test results once the system is built this could be used as a way to measure rider movements for the system to ignore, or be a form of filtering unwanted measurements of the terrain. Near the bottom of the bike and near the pedals a hall effect sensor was proposed to be mounted, with an accompanying magnet being glued to the spinning cranks or pedals to see if the rider is pedalling or not, as if the rider is not pedalling, as optimising pedalling efficiency is not required or possible. The microcontroller will take in the data from these sensors and govern the servos to control the suspension in real-time. The final design followed this initial design mostly, but with some changes. The hall effect sensor for example used more magnets for higher accuracy while the suspended accelerometer was removed as testing showed it was unnecessary.

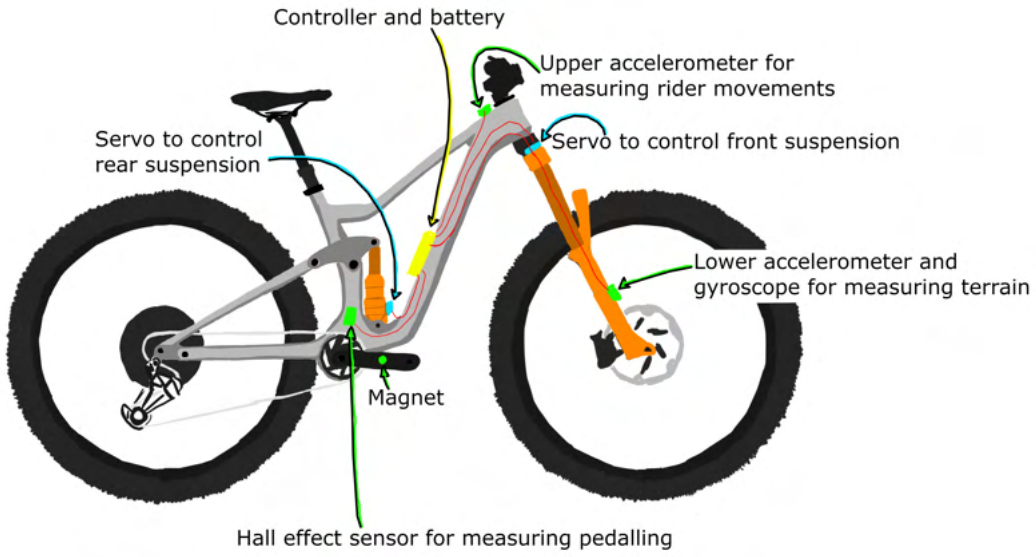


Figure 4: A system overview of the proposed system.

3.2. Component Choice

As the system built was a prototype, easier to use components were chosen. For example, the servos chosen were Tower Pro SG-90 servos. These are not strong, nor waterproof, both qualities desirable in the final system for durability and compatibility. However they are very widely available, inexpensive, use little power, and are well documented. The other components were chosen for similar reasons. The hall effect sensors for example, come preinstalled onto a daughterboard, so a pull up or pull down resistor is already preinstalled and ready for use with the chosen microcontroller.

Some component choices however come at the cost of price. Most parts are not significantly more expensive than what a final system would use, aside from the main controller board. This is because the board chosen, a Raspberry Pi 4, is essentially a single-board computer with an integrated microcontroller. This was chosen as it is far more powerful than a standard embedded controller, and development can be done on the board in real-time with its operating system. This aids in development time and leaves headroom for potentially computationally difficult features to be tested.

In a final and commercial system different components would likely be chosen. Com-

ponents that are robust and as simple and inexpensive as possible would be desired, as the need for prototyping and rapid development would no longer be required. These parts can as a result be bought in bulk on a commercial scale. For example, the hall effect sensor daughter board would be replaced by just the sensor and a suitable resistor. The main microcontroller would also be far less expensive and basic, more typical for an embedded system. An Arduino Nano is a good example of this.

Two tables have been produced seen at Table 1 and Table 2, one covering the costs of this prototype, and one with a rough estimate for more final parts. For the final build plastic or rubber cases would be made for the components to keep out water.

Table 1: Parts cost of prototype system

Parts	Price	Quantity bought	Quantity required	Price per system
Raspberry pi 4	£66.00	1	1	£66.00
Raspberry pi case	£21.15	1	1	£21.15
HDMI adapter	£2.49	1	1	£2.49
SG 90 Servos (Positional)	£7.78	2	2	£7.78
SG 90 Servos (Continuous)	£5.12	2	0	£0.00
RGB LED	£2.25	1	1	£2.25
MPU 6050 accelerometer / gyroscope	£6.20	2	1	£3.10
Hall effect sensor board	£2.29	2	2	£2.29
Hall effect sensor	£2.48	3	0	£0.00
Magnets	£3.99	4	10	£9.98
Push button	£1.99	1	1	£1.99
Bread board cables	£3.25	40	20	£1.63
Bread board Jumper cables	£2.14	40	20	£1.07
Total spent	£127.13		System total	£119.72

Table 2: Estimate parts cost of final system

Parts	Price	Quantity bought	Quantity required	Price per system
Arduino Nano V3	£7.45	1	1	£7.45
Waterproof servo[a]	£15.75	1	2	£31.50
SG 90 servo[b]	£29.99	20	2	£3.00
RGB LED	£2.45	1	1	£2.45
MPU 6050 accelerometer / gyrogyroscope	£3.25	1	1	£3.25
Hall effect sensor	£4.99	6	1	£0.83
Magnets	£6.95	100	4	£0.28
Push button	£1.29	1	1	£1.29
Assortment of resistors	£7.49	400	5	£0.09
10m wire	£3.75	1	1	£3.75
Total spent	£83.36		System total	£53.89

[a] A generic premium and waterproof servo solution.

[b] A budget servo solution that requires waterproofing.

Parts can be bought in bulk for cost reduction.

List does not include casing/waterproofing electronic components.

3.3. System Architecture and Input Process Output Model

An initial system architecture and input process output (IPO) model was made early on in the project, seen in Fig. 5 and Fig. 6. This gives a basic overview of the tasks the system and components must perform during the system's usage. Both cover similar topics, they showcase the system's input, a basic overview of the processing required for functionality, and the system's output. The system architecture diagram also features a rough idea for the algorithm the system will be running in real-time. Essentially the system will have the suspension active or unlocked by default, and only under ideal conditions will it will lock the suspension - if the rider is pedalling over smooth terrain, with the likelihood of the system locking the suspension increasing with inclination.

This ensures the default setting is unlocked, the more forgiving and predictable state for general use in mountain biking.

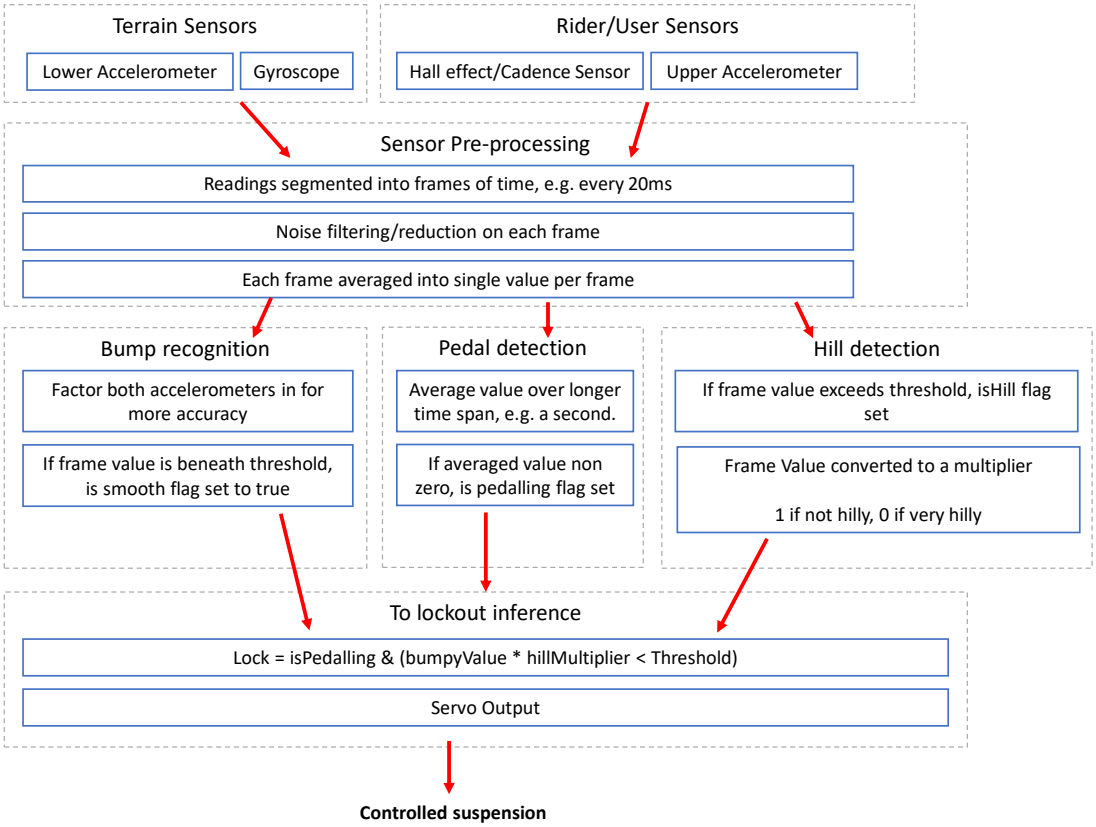


Figure 5: The initial system architecture made.

Input	Process	Output
- Hall effect sensor/If pedals are moving - Gyroscope/Angle bike is at - Accelerometers/Acceleration forces	- Remove noise from sensors - Gauge bumpiness of terrain - Gauge inclination of terrain - Gauge if rider is pedalling - If the rider is pedalling over smooth enough and steep enough terrain, disable the suspension for increased efficiency. - Control front and rear servos	- Servos/suspension enabled or disabled

Figure 6: The initial Input Process Output (IPO) model made.

This final architecture and model changed quite substantially, seen in Fig. 7 and Fig. 8. Firstly as mentioned prior, the upper suspended accelerometer was abandoned, as it did not provide unique data that was useful. This is because rider inputs were quite difficult to discern from bumps, and that rider input would not impact the unsuspended accelerometer reading regardless. Secondly while it was eventually abandoned or its inclusion postponed, inclination sensing was implemented in the architecture and model, and rather as a binary flag it was implemented as a multiplier, so steepness gradually increased the chances of disabling the suspension more and more. Preprocessing and recognising bumps also differed greatly. While other values such as the hall effect sensor and inclination were indeed averaged out, acceleration was not. Binarizing the output

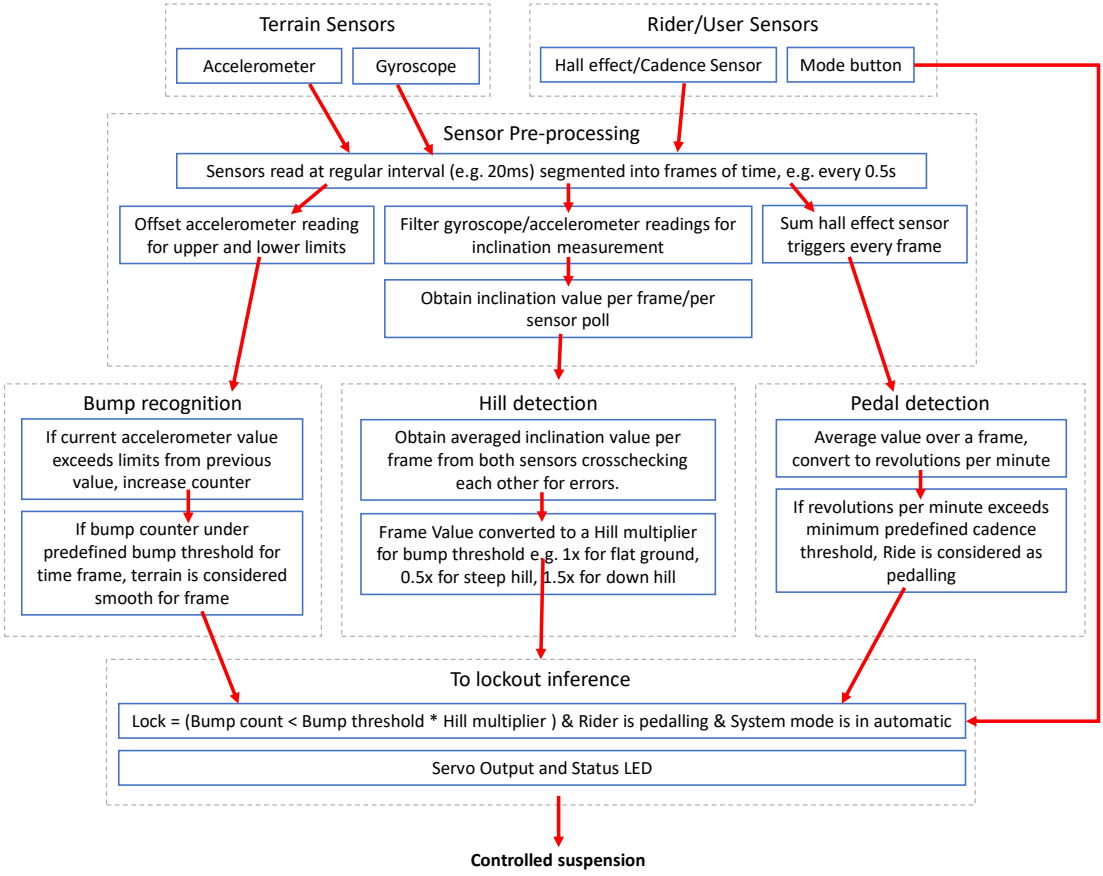


Figure 7: The revised and final system architecture.

Input	Process	Output
- Hall effect sensor/If pedals are moving - Gyroscope/Incline bike is on - Accelerometer/Acceleration forces - Mode button	- Check if system is in automatic mode - Remove noise from accelerometer and gyroscope sensors - Gauge bumpiness of terrain by binarizing accelerometer input - Gauge inclination of terrain from accelerometer and gyroscope - Gauge if rider is pedalling - If the rider is pedalling over smooth enough and steep enough terrain, disable the suspension for increased efficiency. - Control front and rear servos	- Servos/suspension enabled or disabled - Status LED updated

Figure 8: The revised and final IPO.

4. Implementation Overview

4.1. Terrain Measuring

To measure terrain smoothness, an MPU-6050 Accelerometer with integrated Gyroscope was chosen and used. It can measure acceleration in three axes (forward and back for the x-axis, left and right for the y-axis, and up and down for the z-axis) in ranges of $\pm$2G, $\pm$4G, $\pm$8G, and $\pm$16G. The gyroscope can measure rotation around these axes. While the device features eight pins, only four are required, as four are for external devices or expanding functionality. Of the four pins to be used, two pins are for voltage and ground to power the device, and the remaining two are for the Inter-Integrated Circuit (I2C), a form of transmitting data over two pins between more advanced "peripheral" devices (the accelerometer board) with a "controller" device (the Raspberry Pi) (NXP Semiconductors, 2021).

This protocol involves reading (and writing) byte data from (and to) the accelerometer board's registers. Locating correct addresses for configuring the device as well as reading its readings can be achieved with the device's register map. For example, the bits to set the accelerometer's range from the default $\pm$2g to $\pm$16g are bits 3 and 4 at the address 0x1C, seen in Fig. 9 (I2Cdevllib, 2011) (InvenSense Inc., 2013). Once we are here we can send the value "24" to that address. Converting this to binary gives the value 00011000, meaning we have set bits 3 and 4 to high, or to $\pm$16g. This can be seen in Listing 1.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

Figure 9: The register for configuring the MPU6050's acceleration range (InvenSense Inc., 2013).

```
accel_config = 0x1c
# set accelerometer to +-16g
bus.write_byte_data(address, accel_config, 24)
```

Listing 1: Setting the accelerometer configuration register to 24 or $\pm$16g.

4.2. Inclination Measuring

To measure inclination, the accelerometer was again used. Due to gravity being a constant downwards acceleration, one can perform trigonometry on the accelerometer's values to obtain an angle. Doing this gives us a range between -90 and 90 degrees, 0 being when the accelerometer is laid flat.

The gyroscope was not used as it measures rotational change and not tilt or inclination, and deriving this from say an initial value would be prone to drifting.

4.3. Servos

Servos do not behave like motors - their input power or voltage does not govern their behaviour. Instead, they use a pulse width modulation (PWM) signal to tell them where to rotate. For example, the SG-90 servos used feature 180 degrees of rotation, and take in a 20 millisecond or 50Hz binary pulse, where the first 1-2 milliseconds are high, and the rest are low. This can be seen in Fig. 10.

This high period can vary to signal the servo to move to different positions. For example, to move the output shaft to it's -90 degree position, the high pulse would be 1 millisecond. For 0 degrees, 1.5 milliseconds, and for +90, 2 milliseconds. this of course can be more granular for more fine angles. The initial ordered servos that arrived were incorrect and were of the continuous variety. This is where the servo can rotate its output shaft a full 360 degrees and continuously, and where the input pulse simply controls its speed rather than position. This is unwanted behaviour for this project, as the lockout levers require alternating between two fixed positions.

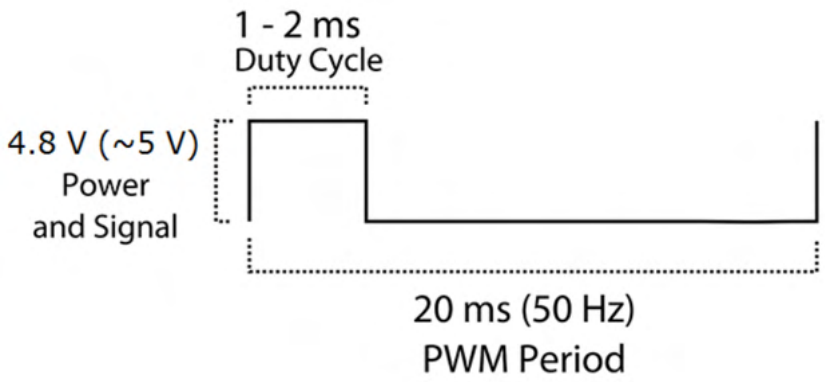


Figure 10: The duty cycle the SG-90 servos use. (Tower Pro, 2014)

Most servos including the SG-90 feature three pins - two for voltage and ground, and the final for PWM. All three were connected to their respective pins on the Raspberry Pi. Using a signal pin for function is an ideal arrangement as if more powerful and power-hungry servos were to be used, they could be powered externally rather than with the Pi.

4.4. Hall Effect Sensor and Button

To check if the rider is pedalling, a hall effect sensor and magnet are ideal. A hall effect sensor measures proximity to magnets so a magnet can be mounted on the bike's pedals and a hall effect sensor mounted on its frame. If the rider is pedalling, the magnet attached the pedal should be constantly passing the hall effect sensor, making the sensor report a pulsing between high and low values. Initially a single magnet was used, but in the final system four were mounted in equal intervals - this increases the accuracy for counting a proper value for revolutions per minute greatly.

The hall effect sensor features four pins. Two for power, voltage and ground, and two for output - one a digital output and one an analogue output. The digital pin was used. Over an interval the number of times the sensor was summed and divided over a set period, giving a value of pedal revolutions per minute. To increment this count a callback was used - whenever the GPIO pin connected to the hall effect sensor's output detected a rise to the high value this would trigger a function call in the program to increase the counter.

A button was implemented into the system, allowing the user to toggle between three

"modes" - one where the suspension is manually locked out, one where it is manually activated, and one where it is governed automatically by the system. The button works in a similar manner to the hall effect sensor output pin - a callback was used to trigger a function call whenever the button went low. Since the button is a simple momentary switch, the way it achieves this is by using a GPIO pin and a ground pin - When the button is pressed, the pin gets connected to ground, lowering its value briefly for the callback to detect. For this setup a resistor should be used, to prevent short-circuiting the pins. In this case an internal resistor was used.

4.5. Implementing the System

4.5.1. Bump Detection

Firstly the accelerometer module and hall effect sensor was set up. With these two working, the program was made to record and log the accelerometer output. The accelerometer was polled for a reading every 20 milliseconds. The output was as expected, however the module defaulted to a $\pm$2g limit. However it was unknown whether this limit was adequate or too low at the time until the system was mounted to the bicycle for real-world data. A method for devising how to quantise the accelerometer data was also developed, to binarize two classes of terrain - bumpy or smooth. Initially, the idea was to simply average the value over a time-frame such as the course of a second, and check whether or not it exceeds a limit to class the terrain as bumpy. Actually doing this on some test data showed however this would not work, as a bump is a sudden change in acceleration - a large increase followed by a large decrease. Averaging out the reading would essentially completely remove all bumps from the data. An example can be seen in Fig. 11 and 12.

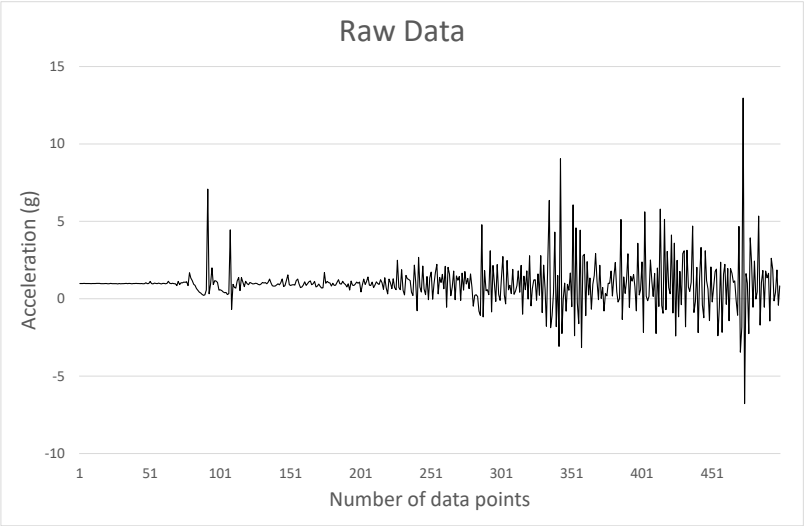


Figure 11: An example of gathered raw accelerometer data.

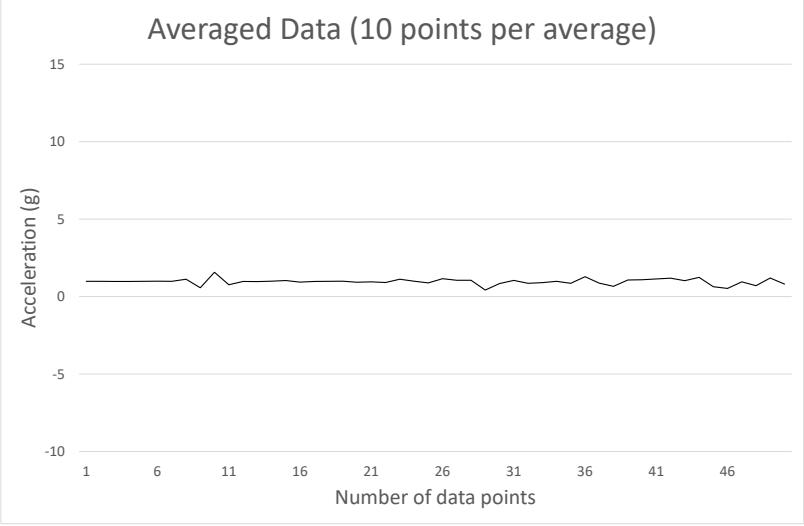


Figure 12: An example of averaging eliminating any useful data.

A second method was then devised. Over the course of a short time-frame, for example, one second, every reading from the accelerometer (polled every 20 milliseconds) would be compared to an upper and lower limit. An example of the upper limit could be 3.5g, a 2.5g increase in acceleration from a bump instead of gravity alone. If the acceleration measured exceeded the limit, a counter was incremented. Then every frame or half-second this count was compared against another threshold, and if it exceeds it, the terrain is considered bumpy during that period. This involved running the program in a constant loop, polling the sensor and performing this binarization. A visual example of this can be seen in Fig. 13.

This method worked and was used in the final system, though it was altered. Only the vertical axis of the accelerometer is used to measure bumps, so inclines in the terrain will cause the settled value to change - on flat, smooth ground the reading will be roughly one g - the force of gravity acting on the sensor. On a steep, smooth hill the reading will be far less. To tackle this the upper and lower limits were flexible and floating - they took the last measured value and would add and subtract a value from it to get constantly updating limit values to use. For example, if the current reading from the sensor was 1.5g, the upper and lower limits would be 1.5g $\pm$ a value. This can be seen in Listing 2.

```
# If current reading exceeds either limit - is a bump
        if (accel_zout_scaled * 9.81) < accelLowerValue or (
    accel_zout_scaled * 9.81) > accelHigherValue:
            bumpCount = bumpCount + 1
        #Update previous reading value
        previousAccel = accel_zout_scaled
```

Listing 2: Setting the previous accelerometer value after gauging if the current reading was a bump.

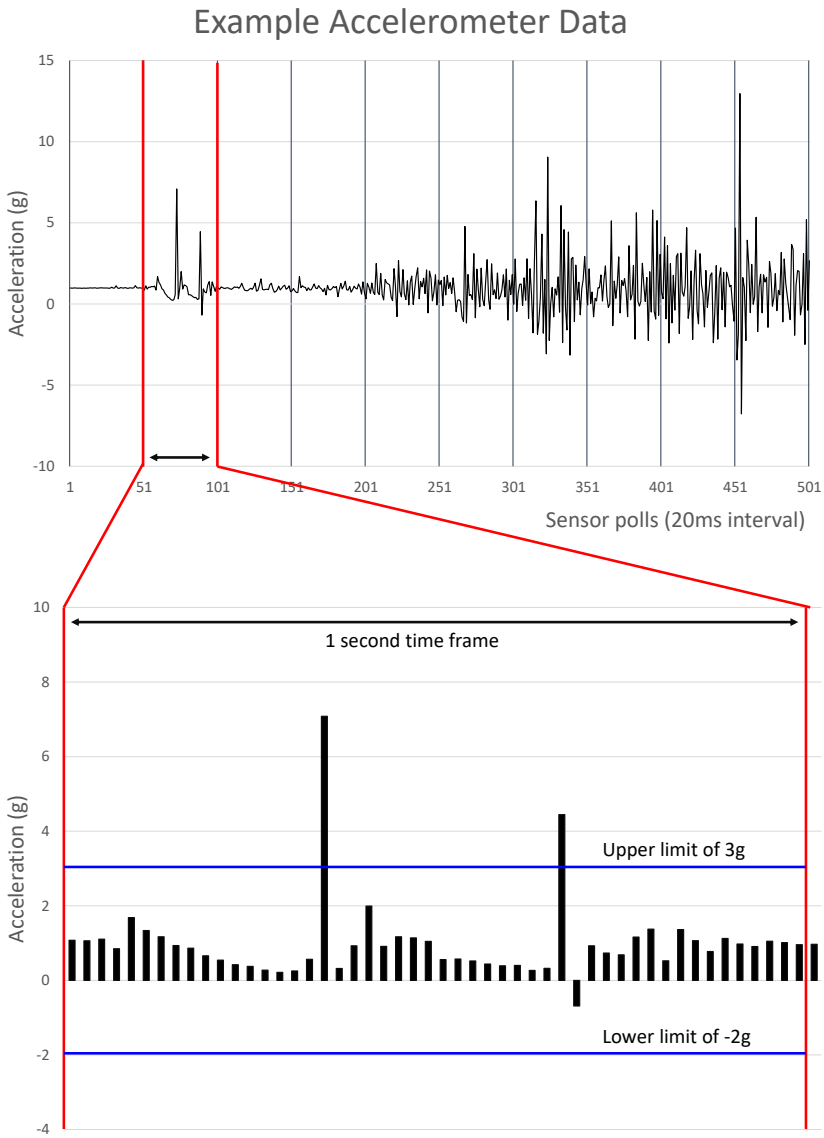


Figure 13: An example of the binarizing method used. During the enlarged time-frame two measurements have exceeded the example 3g upper limit, so two bumps are recorded to have occurred in that time-frame.

The accelerometer was mounted on the bike in multiple positions, with its final position being on an unsuspended portion of the front suspension fork just above the front wheel, seen in Fig. 14.

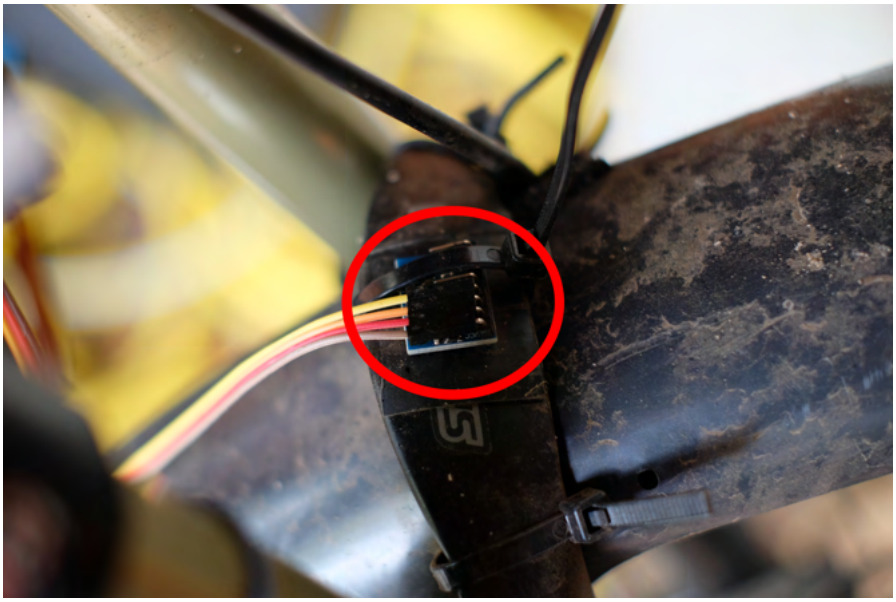


Figure 14: Mounting the accelerometer.

4.5.2. Terrain Inclination Measurement

For measuring inclination in terrain, the accelerometer was again used. It simply triangulates an angle using the force of gravity exerted over the horizontal and vertical axes. When this was first implemented to classify the terrain as steep a simple threshold was used - if the current reading exceeded a set angle the terrain was considered steep. This method of measuring tilt using the accelerometer proved to be ineffective after real-world testing.

4.5.3. Pedalling Detection

The hall effect sensor was quite simple to set up - its sensitivity was adjusted physically using a screwdriver to adjust a potentiometer on its board, and a callback was set up - its counter can be incremented no matter where the program is in execution. Like the accelerometer, it used a counter that was tallied every time-frame, and was compared to

a threshold RPM. The hall effect sensor was mounted near the cranks of the bicycle, and four magnets were glued to the cranks, ensuring the sensor and magnets were aligned. This is pictured in Fig. 15a and Fig. 15b.

(a) The mounted hall effect sensor near the crank arm.

(b) The accompanying magnets. Four were glued to the crank.

4.5.4. Lever Control

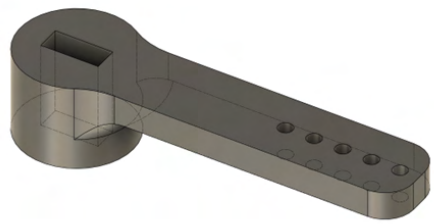
The servos were set up, however the wrong servos were received. They were the continuous version of the SG-90, so providing them with different pulses resulted in the servos rotating continuously at different speeds, rather than adopting different fixed positions. While waiting for the correct models to arrive development continued. Functions were made to move the servos in two different positions - locked and unlocked. The Pi however generates its PWM output through software rather than hardware, so it is imprecise. This causes the servos to jitter constantly if given a pulse to follow, or a position to stick to. This was remedied by changing the functions to have the servos go to the correct position, then simply turning off the servos (giving them no signal) until their position required updating. This can be seen in Listing 3. The final positions the servos would alternate between would have to be measured once they and the system were attached to the bicycle. A form of connecting their output to the suspension would also have to be devised. Initially it was thought that more powerful servos would be required for the final servos, but these inexpensive servos proved to have more than enough torque for this application. Waterproofing would be the only improvement necessary for a final component.

```
def lock():
    global locked
    if not locked:
        #Go to locked position
        servoF.ChangeDutyCycle(2)
        servoR.ChangeDutyCycle(2)
        time.sleep(0.5)
        #Turn off
        servoF.ChangeDutyCycle(0)
        servoR.ChangeDutyCycle(0)
        locked = True
```

Listing 3: The servo locking function.

To attach the servos to the bike the original front suspension lever, pictured in Fig. 16a was replaced by a 3D printed replacement lever, its design seen in Fig. 16b. Then to operate the levers some simple coat hanger wire was used, seen in Fig. 17a and Fig. 17b. For a final version of this suspension lockout lever replacements would be provided, either 3D printed or injection moulded depending on the numbers sold. There are only three major suspension manufacturers, and many share similar parts within their respective brands, so having a range of replacement levers for each lever would be viable. Another possibility could be to replace the levers with gears, though it may be more difficult to set up. Instead of the Jerry-rigged coat wire, a final system could link the levers to the servos using linkage for remote control vehicles.

(a) The original front lockout lever.



(b) The computer aided design (CAD) model of the replacement lever.

(a) The printed lever installed and attached to the servo via a rod.

(b) The rear servo configuration. A replacement lever could be made instead.

4.5.5. System Power, Additions, and Parts Issues

To power the system, a power bank would be used, as the Raspberry Pi uses very little power. More power-hungry components such as bigger servos could directly be powered by the power bank rather than the Pi, as power is separated from their function, as the servos use a PWM signal to determine which position to go to. However for the parts selected in the prototype in testing this was not needed. The project then faced an issue and was put on hold as the power bank obtained for this system broke, and shipping time for a replacement (and replacement positional servos) took a long time. During this downtime a button and LED were introduced to the system. The button served to give the user a manual override, allowing them to toggle between manually unlocked or active suspension, and have it be governed automatically. The LED would indicate which mode the system was currently in - green for auto, blue for active, and red for locked. The button was implemented like the hall effect sensor and the servos - when pressed it would trigger a callback function, and the colour was changed by setting each colour channel's PWM duty cycles to 0% or 100% in combination. In this callback function a flag was set for which mode the system was in, so it could swap modes while still executing the loop.

The inclination measurement was also altered. Instead of being a binary result, the reading was changed to be a granular multiplier - if the angle was negative (going down a hill and most likely coasting) the greater the less likely the system would lock the suspension, while if the angle was positive (going up a hill), the system would be more likely to lock the suspension. This was done by simply taking in the reading for angle

(between -90 and +90 degrees), offsetting the value to be between 0 to +180 degrees, then dividing by 90. The number of bumps threshold for bump detection is then multiplied by this value, seen in Listing 4. This essentially turns the value into a granular multiplier - a negative angle (downhill), will reduce the threshold making it less likely for the system to lockout the suspension, while flat ground is neutral and uphills will make the system favour lockout. The multiplier factor could be tested and modified, though due to unforeseen issues this would not take place.

```
# Turning rotation into a multiplier
            # Average angle over a frame
            angleTotal = angleTotal / outputIntervalCount
            angleMultiplier = angleTotal + 90
            #-90 (downhill) becomes 0, 0 (flat) becomes 1, 90 (uphill
   ) becomes 2
            angleMultiplier = angleMultiplier / 90
            bumpThreshold = bumpThreshold * angleMultiplier
```

Listing 4: The frame averaged angle value is normalised into a multiplier for the bump frame count threshold.

The replacement power bank arrived and the system was mounted to the cycle, in a frame bag, seen in Figures 18a, 18b, 19a and 19b. The components were zip-tied to their respective locations, with rubber in-between the cycle and the component to keep them securely in place. In a final design zip-ties would likely be used again, as it is a common semi-permanent fastener used in the sport, but the components would likely feature compact cases to keep them protected from the elements, and for mounting security. The main controller would likely be much more compact also, allowing for frame mounting. For a final system a power bank would be quite optimal, as they are inexpensive, feature battery management circuitry, and can charge other devices riders may have for additional functionality. A final power bank would be far smaller than the one used, as the used power bank was very large and heavy, and a smaller power bank would still provide adequate run time. The button and LED would also be handlebar mounted for ease of access and visibility.

(a) The bike with the system installed in the bag and on the bike.

(b) Wiring can be significantly tidied by rerouting them with longer wires.

(a) The Pi in the frame bag. The power bank is on the other side.

(b) The mode button and mode/status LED, currently green.

4.5.6. Gathering Data and Initial Thresholds

real-world test data was then gathered. To record this, a dedicated program was written to do so to streamline the process. It recorded all sensor input and used the momentary button as a start-stop button, and the LED as a recording indicator. Differing terrain types were recorded, including smooth concrete, bumpy grass, mildly bumpy gravel, and a commute to campus grounds with mixed terrain. These were repeated for a second mounting position for the accelerometer. The first position was just above the front wheel, on an un-suspended part of the bike, while the second was near but on a suspended part of the bike. The first set of data had to be discarded however, as at this point in the time the accelerometer was using its default limits of $\pm$2g. A second run

was performed using a limit of $\pm$16g which provided useful results. Firstly, it decided where the accelerometer would be placed - the suspended location did indeed reduce impacts over larger bumps. While bumps could still be detected in the suspended position, the position was ruled out as the data can be influenced by the suspension being locked or active - if limits were set for a suspended position, when the suspension is locked it would likely exceed these values more often than before. A comparison between the two positions can be seen in Fig. 20.

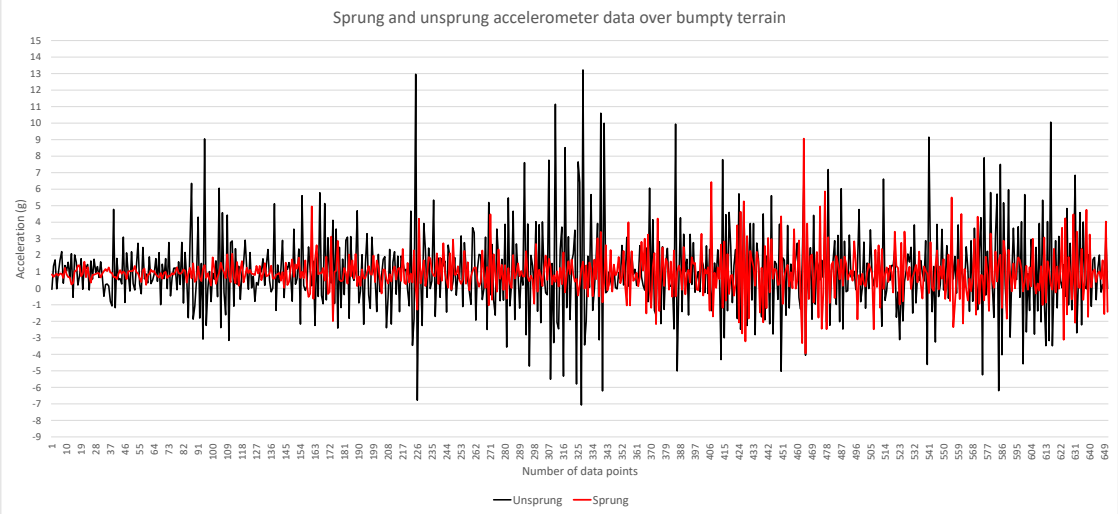


Figure 20: A comparison between suspended and unsuspended mounting positions. Suspended data still shows bumps, but at a smaller scale.

From the test data seen in Fig. 21 initial values for bump definition and thresholds were devised. The values should be very conservative - the suspension should favour being active rather than locked. This is to provide a reliable and consistent experience and to err on the side of caution, with the setting with the most traction and comfort being the default. An acceleration change of 3.5g was chosen to define a bump, with bumps at a frequency of 0.64 bumps per second deemed as the threshold for classifying something as bumpy terrain. These values aim to roughly feature gravel or slightly bumpy terrain as the threshold, as the suspension provided by the bicycles tyres are sufficient for this type of terrain. These values should be honed and refined with further testing.

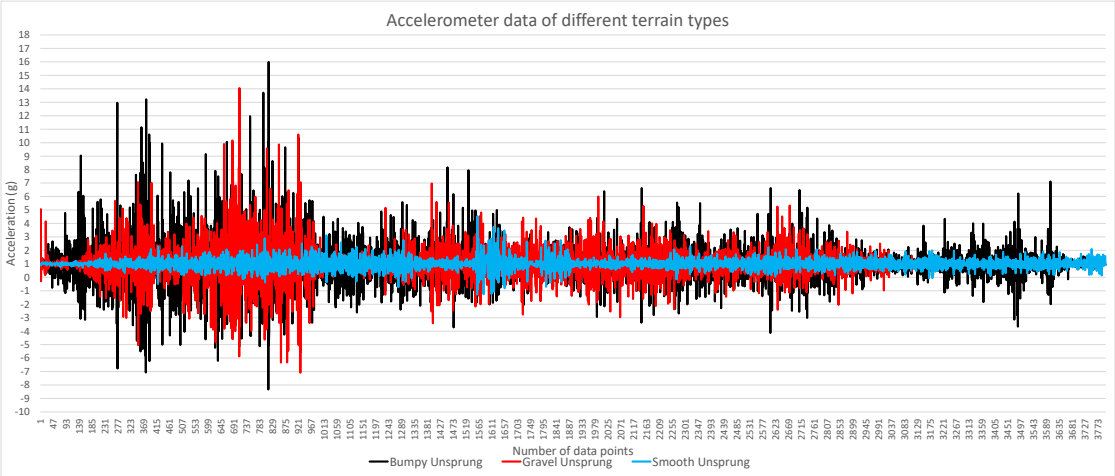


Figure 21: A comparison between different terrain types. The system should aim to have gravel as a threshold on what is considered bumpy.

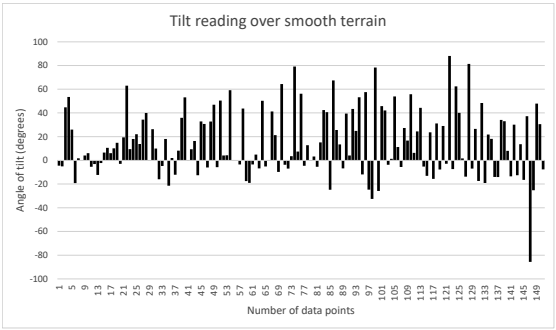
4.5.7. Sensor Polling and Frame Length

Values for the frame length and polling period was also finalised. The sensors would be polled every 20 milliseconds and each frame would be 0.5 seconds long. This in theory could easily be shortened for quicker response times, but in actuality the servos take 0.5 seconds to move between positions, so even if it could it would not change the output. A minimum frame length would also require being long enough to get enough resolution to average reading of values such as the hall effect sensor. For example, if the frame length was 20 milliseconds and the polling interval 10, the final reading after being averaged can only be 0, 1 or 0.5, so thresholds in-between those values would be ineffective.

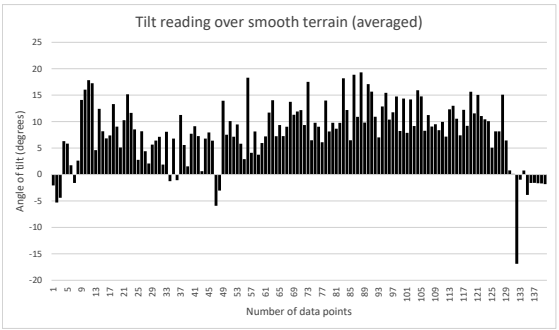
4.5.8. Inclination Issue Encountered

It was here that it was discovered that measuring tilt would not be feasible for this project. The real-world data showed that during riding over rough terrain, the angle value fluctuated wildly and greatly, and was essentially unusable. It was thought that since averaging the acceleration readings eliminated bumps seen previously the same could be done for this value, but this did not rectify this issue. In hindsight, the cause is quite obvious. The bumps are providing sudden values of acceleration in random directions, so it is no longer possible to triangulate the angle gravity is acting on the

sensor. A brief look at possible solutions show that this feature is unlikely to make a swift return - specifically vibration tolerant sensors are very expensive and specialised. In the future this can potentially be re-implemented with the existing accelerometer and gyroscope module rectifying the issue by using the two inputs, combining and filtering their input with methods such as the Kalman method (Alfian et al., 2021) (ST, 2020). This works as the sensors can essentially cross-reference each other to validate their readings. For example, if an external force such as a bump causes the accelerometer read a sudden change in angle, the gyroscope can be used to confirm whether or not the sensor module experienced any actual change in angle or rotation, making the reading tolerant to vibration. Multiple sources show and have their own commercial modules rated as vibration tolerant (ST, 2020) (TE Connectivity, 2020). Due to this issue being discovered late into the project its solution was not implemented. In Figures 22a and 22b this lack of vibration tolerance can be seen, even with test data from traversing over smooth terrain.



(a) The reading fluctuates greatly, even on smooth terrain.



(b) Averaging the reading over the course of longer periods do not solve the issue.

4.5.9. Internal Documentation

As the system aims to be highly expandable in the future, the code base is clearly and thoroughly commented. This ensures in the future the system can be well understood and worked on. This includes register addresses, function descriptions, and sections of the program are clearly divided. An example of this can be seen in Listing 5.

```
############################################################
    #VAR INIT
############################################################
averageCount = 0 #Every poll increments counter
accelTotal_x = 0 #Sum of accel x to average over frame
accelTotal_y = 0 #Sum of accel y to average over frame
accelTotal_z = 0 #Sum of accel z to average over frame
updateInterval = 0.02 #in seconds. The polling interval
outputInterval = 0.5 #in seconds. How long a frame is
outputIntervalCount = outputInterval / updateInterval

bumpCount = 0 #Sum of bump count to average over frame
hallEffectCount = 0 #Sum of hall to average over frame
angleTotal = 0 # To average out angle value as it's quite erratic
    with bumps

bumpThreshold = 0.5 #How many bumps in an interval
bumpLimit = 3.5 * 9.81 #in m/s/s Previous g value +- this in m/s/s
cadenceThreshold = ((40 * 4) / 60) * outputInterval # Minimum
    pedalling RPM.
angleThreshold = -5 # In degrees, 0 is flat, -ve is downhill. Sorta
    deprecated.
previousAccel = 1.0 # The previous acceleration value you compare the
     current one to
############################################################
    #END OF VAR INIT
############################################################

```

Listing 5: Variable initialisation

5. Evaluation

5.1. Measuring Performance

5.1.1. Evaluating Efficiency Savings

Evaluating this project can be done in several ways. Unfortunately one of these metrics cannot be measured - the actual efficiency benefits from the suspension being locked out. A few previous papers conclude there is no difference in times or output riding with locked or active suspension, putting the entire premise of lockout and this project in jeopardy. However a paper review (Nielens and Lejeune, 2004) concludes that most papers find that suspension is beneficial for rough terrain, but that waters muddy when looking at pedalling efficiency. The majority of papers look into climbing over rough terrain rather than smooth terrain, where any inefficiency can be counteracted by the benefits of suspension. Many papers also do not test cases where suspension could have a greater hindrance to efficiency, such as standing up to pedal (Nielens and Lejeune, 2001), or ignore potential areas of inefficiency with ideal testing and modelling. The topic is very scarcely researched also, especially factoring smooth terrain.

The majority of papers also focus on comparing whether or not featuring rear suspension on top of the typical front suspension is faster for racing. Given results are fairly split it can be safe to assume a system that allows for the bicycle to essentially act as if it has no suspension or be fully suspended whenever either is ideal is an improvement to only having one or the other.

As for testing the system in this field myself, it is unfortunately not feasible. One traditional method of measurement would be a power meter at the cranks/pedals of the bike, and timing results at a given wattage. This would not provide useful results, as riding at a fixed wattage for example will not measure the bodily exertion required to maintain that power, which is the true goal of suspension lockout to improve. Measuring bodily exertion is also not feasible, as for example measuring VO2 Max would require riding over multiple types of terrain with a non-portable measurement machine.

5.1.2. Evaluating Product Cost

Other areas must be looked at then. In terms of cost, the system is a great success. While Table 2 does not include all costs such as component cases and servo linkage, the

total cost of a final system would remain under £100. It provides users with electronically controlled suspension for a fraction of the cost of the commercial systems and can be adapted to existing suspension designs. This is orders of magnitude less than commercial offerings (Cunningham, 2018), and does not require a change in suspension components. However, the low cost pits the system against manual controls, such as handlebar remote lockout solutions. This I think is acceptable, as the system can grow in capability tremendously and for little cost - instead of a simple binary output for lockout for example, it could be made to control damping or other more advanced suspension features remotes cannot. Potential expansion such as wireless communication between parts and more sensor input could also be added. Another great area for expansion is combining the system with a bicycle trip computer, displaying speed, measuring times, as well as more involved features such as navigation and power reading. This can then place the system in the field of trip computers, boosting its value greatly, as for the same cost as a trip computer, automatic suspension can be had included with one.

5.1.3. Evaluating Other Metrics

As for metrics, the final system's response time is 0.5 seconds, as this is as fast as possible while still allowing the servos to move to their positions adequately. Should this limitation be improved with faster servos, the length of a frame and therefore the system's response time can be shortened easily, though only to a point to allow for enough decimal precision for averaged values as previously mentioned in subsection 4.5.7. In use the current response time is adequate - the system changes the state of the suspension faster than the rider can react, let alone physically change the suspension manually, though this does not consider a human's ability to change the bike's state preemptively. For constantly changing terrain types the system is an ideal solution.

In terms of classifying terrain, the system did acceptably. With the previously mentioned values bumpy and smooth terrain was suitably binarized. The values can also easily be refined with further testing. To expand the system further, more complex classification methods could be used, such as Hidden Markov Models (HMMs). This would be suitable unlike something like Deep Learning Models, as the HMMs can be trained sufficiently in advance, and are easier to run, being more likely to run on embedded hardware. Engineering features for the model would also not be difficult, eliminating

the key strength of the machine learning route. Reintroducing inclination measurement would be a great addition to the system as well, as it fills the system's last blind spot of environmental input.

In terms of ease of use, on the software side the system is a success. All a user has to do is turn the system on and off, with manual controls as a simple optional process. If more features were added a small dot-matrix screen or companion app could be added, though this would add potentially harmful complexity and cost. This though as previously mentioned can be combated by adding bicycle computer functionality, changing the system to being a bicycle computer that also features automatic suspension. As for physically installing the system, the prototype was not ideal, requiring a bicycle frame bag for the large controller and power bank. Ideally the board should be very small, and powered by a small battery pack like the commercial systems. Other components such as sensors are currently acceptable and ideal - a simple zip-tie is familiar to cyclists and works well. The servos in a final system ought to have an easy way of attaching to the bike, and generic/universal methods of turning the levers. The system would also obviously have to be waterproof, something easily doable by making small plastic or rubber cases for each component. An alternative approach to the bicycle trip computer and its additional functionality would be to create independent units for the front and rear suspension, like the RockShox Flight Attendant system. This eliminates wires, though increases complexity and cost, by requiring wireless communication and separate batteries. A potential basic version of the system could be made instead, where only accelerometer data is used and each system cannot communicate with each other.

5.1.4. Evaluating Project Management

In terms of the project and its management, I think it was done well. Instead of being as ambitious as possible, a realistic, highly expandable foundation was built, giving adequate time for setbacks and experimentation experienced. If I were to do this project again I think I would attempt to implement HMMs for terrain classification, though if factoring in the setbacks experienced I don't think I would have had the time to do so. A dot-matrix screen could've been a more attainable extra, as well as looking into bicycle computer functionality such as speedometer and stopwatch functionality, and inclination measurement would be a must.

The final Gantt chart shown in Chart A shows that the project followed the initially

planned schedule from the progress report Gantt chart (shown in Chart B) quite well, up until the setbacks where the power bank broke and the servos required replacing. After this the project followed the initial chart well again, though offset by a few weeks.

6. Conclusion

In this report the process of designing and building a successful embedded system from the ground up was demonstrated and documented. Controlling mountain bike suspension lockout has successfully been automated electronically, with a system that is affordable and widely compatible with a range of suspension components. Methods of devising real-time terrain monitoring with an accelerometer, rider monitoring with a hall effect sensor, and reactive output by servos was completed and documented. The methods used are simplistic and effective, and can be easily expanded and added to or changed to accommodate more sensors or more advanced techniques. Experiments such as sensor placement and typical terrain readings have been performed, and methods to deduce inclination or tilt sensing tolerant to vibration have also been researched. Some limitations are present, mainly due to time constraints. A final system for commercial sale would feature lower-cost components packaged in a smaller form than the prototype, and could use more sophisticated methods for terrain classification, such as HMMs. Tilt sensing would also be re-implemented.

References

Alfian, R. I., Ma'arif, A., and Sunardi, S. (2021). Noise reduction in the accelerometer and gyroscope sensor with the kalman filter algorithm. *Journal of Robotics and Control (JRC)*, 2(3).

Boonstra, M. C., van der Slikke, R. M., Keijsers, N. L., van Lummel, R. C., de Waal Malefijt, M. C., and Verdonschot, N. (2006). The accuracy of measuring the kinematics of rising from a chair with accelerometers and gyroscopes. *Journal of Biomechanics*, 39(2):354–358.

Cunningham, R. (2012). First look: Rockshox/lapierre show electronic shock at morzine. *https://www.pinkbike.com/news/ei-and-RockShox-Show-Electronic-Shock-2012.html*.

Cunningham, R. (2018). Review: Fox live valve suspension. *https://www.pinkbike.com/news/review-fox-live-valve-suspension.html*.

I2Cdevllib (2011). Mpu-6050 6-axis accelerometer/gyroscope: I2c device library. *https://www.i2cdevlib.com/devices/mpu6050#registers*.

InvenSense Inc. (2013). Mpu-6000 and mpu-6050 register map and descriptions revision 4.2.

Kazimer, M. (2021). Review: Rockshox' new flight attendant suspension system. *https://www.pinkbike.com/news/review-rockshox-flight-attendant.html*.

Kolotoff, V. (2015). Electronic mountain bike suspension control system. *https://habr.com/en/post/158449/*.

Luinge, H., Veltink, P., and Baten, C. (1999). Estimating orientation with gyroscopes and accelerometers. *Technology and Health Care*, 7(6):455–459.

Nielens, H. and Lejeune, T. (2004). Bicycle shock absorption systems and energy expended by the cyclist. *Sports Medicine*, 34(2):71–80.

Nielens, H. and Lejeune, T. M. (2001). Energy cost of riding bicycles with shock absorption systems on a flat surface. *International Journal of Sports Medicine*, 22(6):400–404.

NXP Semiconductors (2021). Um10204 i2c-bus specification and user manual.

Ramsden, E. (2006). *Hall Effect Sensors Theory and Applications by Edward Ramsden*. Elsevier.

Simon, D. and Ahmadian, M. (2001). Vehicle evaluation of the performance of magneto rheological dampers for heavy truck suspensions. *Journal of Vibration and Acoustics*, 123(3):365–375.

ST (2020). Precise and accurate tilt sensing in industrial applications. *https:// www.st.com/ resource/ en/ application_note/ an5551-precise-and-accurate-tilt-sensing-in-industrial-applications-stmicroelectronics.pdf*.

TE Connectivity (2020). *https:// www.te.com/ usa-en/ industries/ sensor-solutions/ insights/ tilt-sensors-white-paper.html*.

Tower Pro (2014). Servo motor sg-90 data sheet. *https:// datasheetspdf.com/ pdf-file/ 791970/ TowerPro/ SG90/ 1*.

Tyler, B. (2012). Hacktastic! diy automatic electronic suspension lockout w/ stealth remote buttons! *https:// bikerumor.com/ hacktastic-diy-automatic-electronic-suspension-lockout-w-stealth-remote-buttons*.

Wang, M., Zuo, L., Yang, Y., Yang, Q., and Liu, T. (2014). Complex terrain perception based on hidden markov model. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1468–1473.

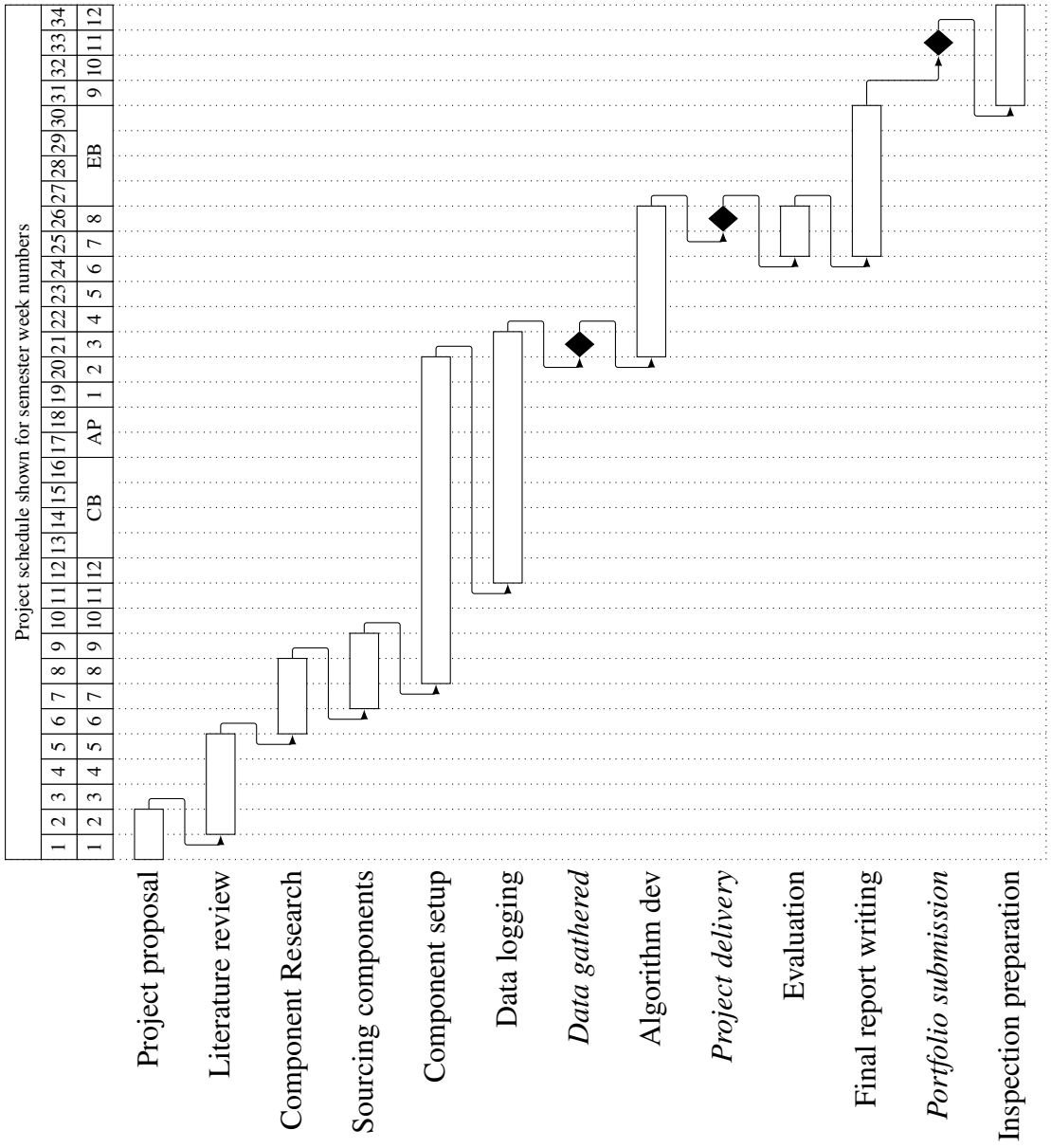
Welch, G. and Bishop, G. (2001). An introduction to the kalman filter. *Proc. Siggraph Course*.

Welch, G. and Bishop, G. (2006). An introduction to the kalman filter. *Proc. Siggraph Course*, 8.

Wolf, D., Sukhatme, G., Fox, D., and Burgard, W. (2005). Autonomous terrain mapping and classification using hidden markov models. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, volume 2005, pages 2026–2031.

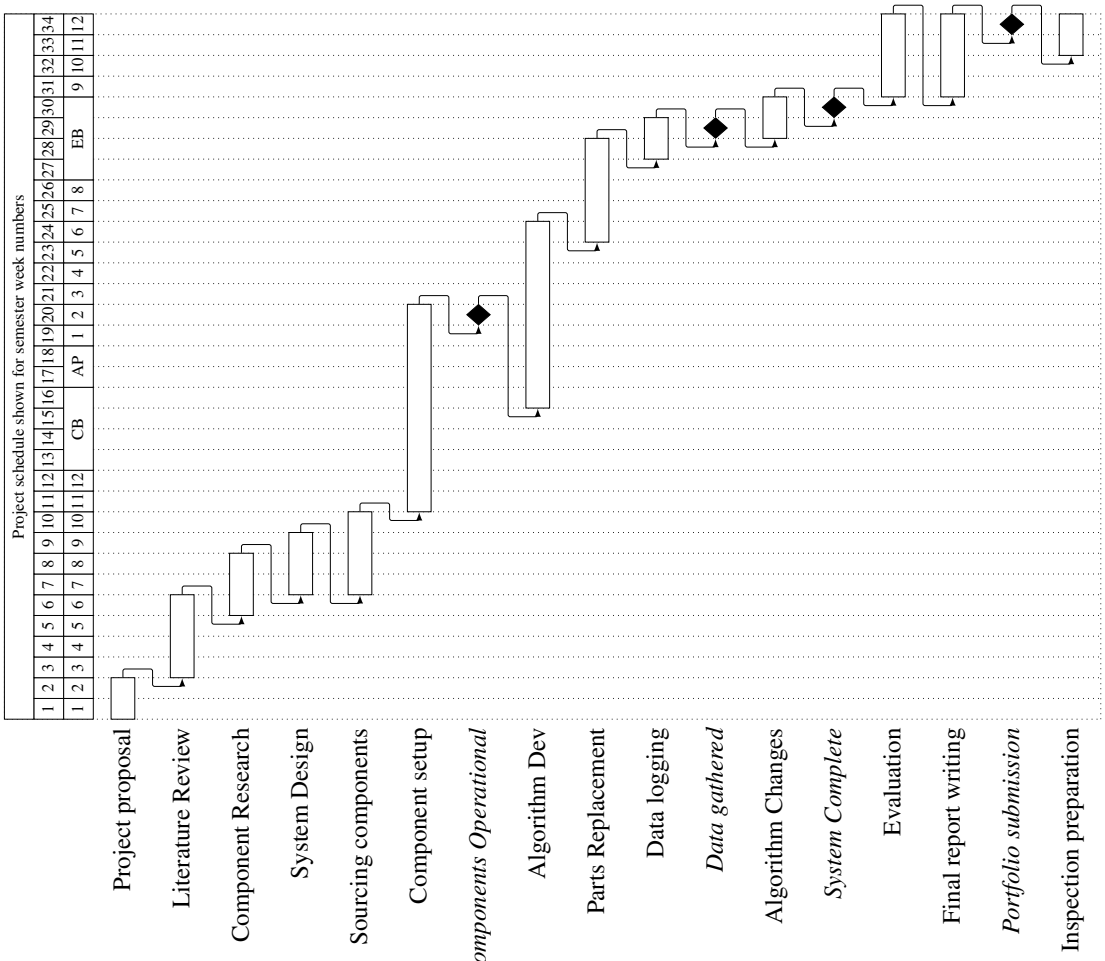
CMP-6013Y

A. Initial Gantt Chart



Reg: 100242165 44

B. Final Gantt Chart



Reg: 100242165 45