Self-Balancing Robot

Third Year Individual Project – Progress Report

Nov 2016

Abdul Gafar

9097951

Supervisor:

Dr. Joaquin Carrasco Gomez

School of Electrical and Electronic Engineering

Contents

1. Introduction and Motivation

Self-balancing robots have sparked interest of many researchers, students and hobbyist worldwide. From an engineer's perspective, it is an inverted pendulum on wheels. The inverted pendulum is a classical problem in control systems due its unstable nature. To the average individual, one of the triggers for the curiosity towards the self-balancing robots was the release of the Segway PT (Personal Transporter). These robots became very popular because of their manoeuvrability, in particular their short turning radius [1]. The Segway has been used in many industries, from tourism in the park, police, and even ambulances. In recent times, a derivative of the Segway, the hoverboard, has been in the headlines of social media, once again directing the attention of many towards the engineering behind.

In any balancing robot knowing the tilt angle is critical, thus an inertial measurement unit (IMU) is a necessity. The IMU is predominantly composed of a gyroscope and an accelerometer. Both sensors have their advantages and disadvantages, therefore to obtain a more accurate measurement the data has to be fused. As part of the project, a technique known as Kalman filtering will be explored. If implemented and tuned correctly, the Kalman Filter "is the best possible (optimal) estimator for a large class of problems." [2]

As a Mechatronics student, making a self-balancing robot is the ideal project. The core of the project is control, thus it will allow the application what has been covered to date and exploration of new material such as alternative controllers, data fusion or odometry. In addition, the project is sufficiently broad to refine knowledge in the areas of embedded systems, programming, PCB and mechanical design. The material to be covered has a broad range of applications, developing many skills transferrable to future projects.

The purpose of this report is to outline the plan of the project and to summarize the progress achieved to date.

2. Aims and Objectives

The aim of the project is to design, make and program a Self-Balancing Robot with a self-developed Kalman Filter. In order to successfully complete the project, the following objectives need to be met:

- Perform Literature review on Kalman Filters and implement in MATLAB
- Develop a Kalman Filter to fuse data from the gyroscope and accelerometer
- Design and assemble the chassis of the robot
- Develop a PID controller to enable the robot to stay upright

If time permits, the list below outlines the possible additional targets:

- Explore the use of a LQR or Fuzzy Logic controller
- Create a remote controller for the robot
- Improve the control algorithm to be able to support loads including asymmetrical loads
- Create Autonomous Pre-programmed paths using odometry

3. Existing Work

Balancing Robots have existed for several years, thus many papers and theses have been written about them. Some are purely for learning purposes, as is the case. Others are to research the application of certain theory such as the LQH controller or fuzzy logic. And certain theses, aim to develop a robot for a specific purpose, this includes a butler robot or an interactive balancing robot to be used in exhibitions.

In most cases, students would focus on a certain aspect, such as data fusion, analysis of dynamics or controller design, and the rest of the robot would be built using simpler techniques. For example, they would focus on using a Kalman filter and use a PID controller or focus on LQR controller and use a Complementary filter.

For sensor fusion, the complementary filter and the Kalman filter are the most commonly employed techniques. The Kalman filter will be further explained in section 4. The complementary filter, simply consists of a low pass filter for the gyroscope and high pass filter for the accelerometer. Whilst, the Kalman filter is accepted as the best estimator, in a specific case the complementary filter appeared to perform better. [3]

To maintain the robot upright, the commonly mentioned controllers are Proportionalintegral-derivative (PID) and the Linear Quadratic Regulator (LQR). A Linear Quadratic-Gaussian controller has also been tested, however, due to a slow microcontroller, it was not successful. [1] In a more complex situation, where the robot also moves around, two controllers are used. For example an LQR controller to balance the robot and a PID controller to control yaw. [4]

4. Kalman Filter

The Kalman Filter (KF) was first introduced in 1960 by Rudolf E. Kalman [5]. Since then, due to its adaptability and usefulness, research and development has continued creating variants such as the Extended Kalman Filter or the Unscented Kalman Filter [2]. The KF was famously used in the Apollo program, ultimately taking Neil Armstrong to the moon [6]. “The KF is over 50 years old but is still one of the most important data fusion algorithms in use today [7].” Its use ranges from navigation and object tracking to investment banking and economics.

Data fusion in essential in this case due to the nature of the gyroscope and accelerometer. The accelerometer measurements are more susceptible to noise, whilst the gyroscope drifts over time. This makes the accelerometer readings more accurate in the long run, and the gyroscope more accurate over a short space of time [8]. To resolve the dilemma the KF can be used.

In addition to the accuracy of estimation, the KF is appealing because it is a recursive method. The current state is dependent on the previous state, which means that not all the data is necessary, allowing it to be implemented in a simple microcontroller without large storage [9]. One of the barriers for the use of the KF is difficulty in understanding due to the lack of standard notation.

4.1. Creating a Model

To implement a KF, the system needs to be modelled in state-space form. The difference equation (1) that can be used to represent the process state and equation (2) models the measurements [2].

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (1)$$

$$z_k = Hx_k + v_k \quad (2)$$

Where [6]:

x_k is the state vector, contains variables to be estimated i.e. angle or bias

u_k is the vector containing control inputs i.e. angular acceleration

A is the transition matrix, which maps the state parameters at *k-1* to *k*

B is the control input matrix, maps the controlled inputs u_k to the state vector

z_k is the measurements matrix

H is matrix that transforms the state vector into measurements

w_k and v_k are the vectors containing the process noise and measurement noise respectively. The noise is assumed to be zero mean Gaussian distributed with a covariance Q and R, respectively i.e. w_k~ (0, Q) and v_k~ (0, R).

4.2. The Kalman Filter Algorithm

The KF is composed of two sets of equations, time update and measurement update equations.

4.2.1. Time Update

The following equations describe the time update stage, also known as the prediction stage:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \quad (3)$$

$$P_{k|k-1} = A_k\, P_{k-1|k-1}A_k^T + Q_k \quad (4)$$

Where:

$\hat{x}$ is the state estimate

P is the process covariance matrix

It is important to understand the subscript. **a | b** means **a** given **b** and all previous states before **b.** For example $\hat{x}_{k|k-1}$, is the estimate at k based on k-1 and on all the states before k-1. $\hat{x}_{k|k-1}$ is known as the priori state, $\hat{x}_{k-1|k-1}$ is the previous state and $\hat{x}_{k|k}$ is the posteriori state.

4.2.2. Measurement Update

The following equations are used in the measurement update:

$$K_k = P_{k|k-1}H_k^T(H_kP_{k|k-1}H_k^T + R_k)^{-1} \quad (5)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1}) \quad (6)$$

$$P_{k|k} = P_{k|k-1} - K_kH_kP_{k|k-1} \quad (7)$$

Where: K is the Kalman Gain Matrix

4.3. Overall Diagram

The KF runs in a loop shown in the diagram below:

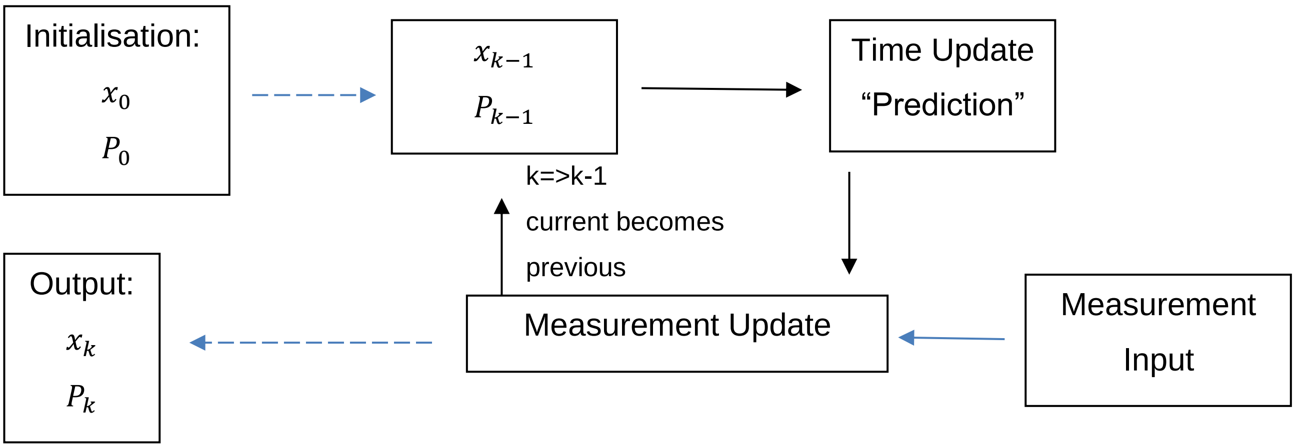

Figure 1 - Kalman Filter Loop (Diagram adapted from iLecture online [22])

4.4. Kalman Filter Practice in MATLAB

In order to better understand how KFs are implemented, examples were done in MATLAB. The first example was following a tutorial, which the 'real' measurement was a constant voltage [10]. In the tutorial the computation was shown, but no code was given. Implementing it MATLAB helped visualize how the KF can be realised in code. The MATLAB code can be found in Appendix 4. The figure in the following page shows the output:

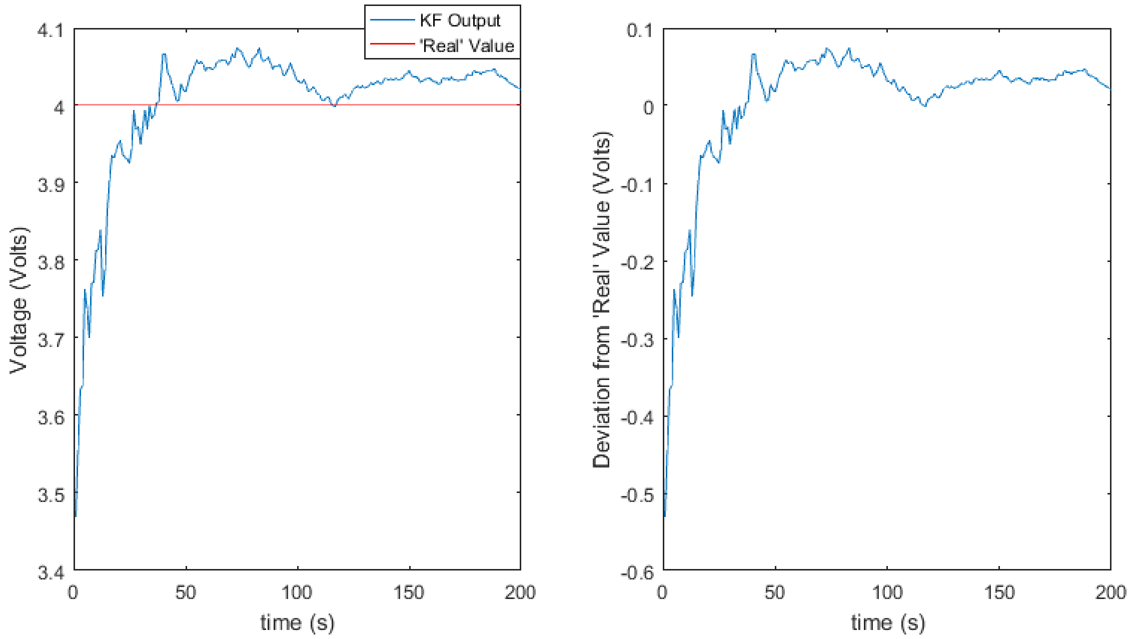

Figure 2 – Output of Kalman Filter implement in MATLAB for a constant voltage

To further aid understanding, a simple example was created and implemented. It consists of measuring the displacement of an object travelling in 1-D at a constant velocity of 1.5m/s. The MATLAB code can be found in Appendix 5**.** The figure below shows the output:

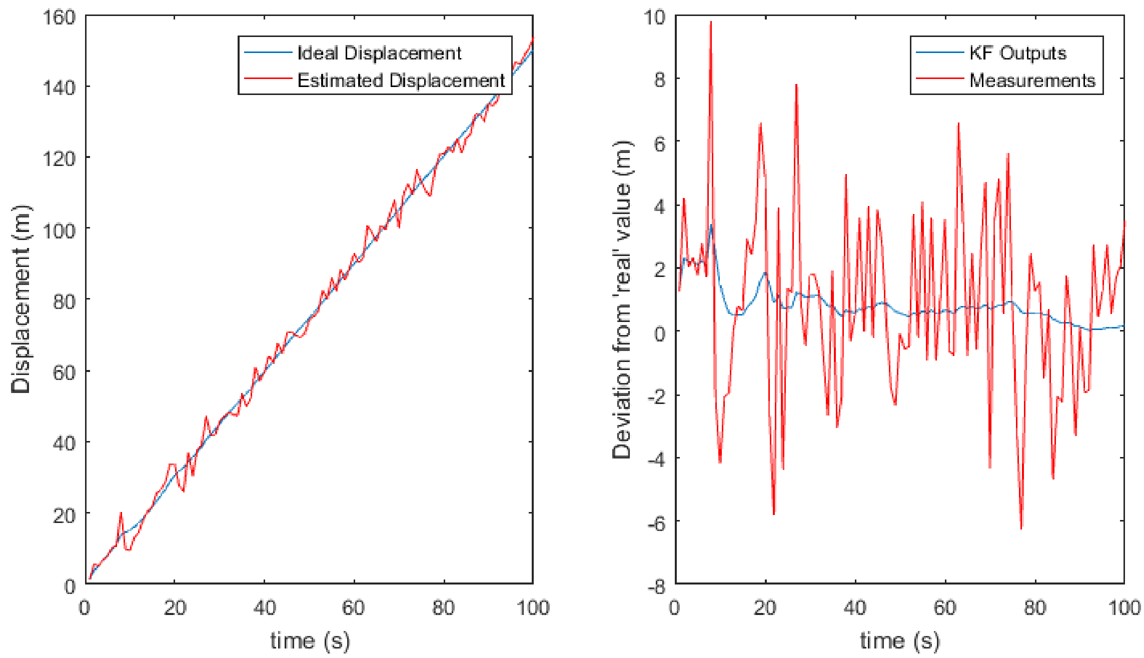

Figure 3 – Output of Kalman Filter for an object traveling away from origin at 1.5m/s

5. Hardware

5.1. Microcontroller

The microcontroller chosen was the Arduino Uno. It has a relatively small footprint, keeping the robot compact. The main advantage of the Arduino is large community and extensive collection of libraries, if any problems are stumbled upon, there is a higher chance that someone else has found a solution.

5.2. Motors

In order to establish the motors required, a calculation of the required torque is necessary. The diagram to the right shows a sketch of the balancing robot.

$$\tau = \|\boldsymbol{r}\|\|\boldsymbol{F}\| sin\theta \quad \text{...} (1)$$

Where: τ is magnitude of the torque, **F** is the force vector, **r** is the position vector and θ is the angle between force and position vectors.

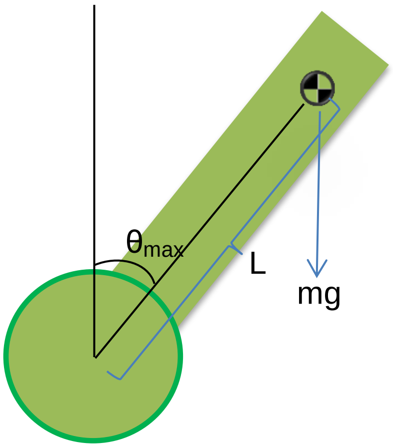

Figure 4 – Force due to gravity

Assuming the distance between the pivot point and the centre of mass (L) is 12cm, the maximum tilt angle (θ_{max}) is 40° and the mass of the robot (m) is 0.7kg.

$$\tau = L * mg * sin\theta = 0.12 * 0.7 * 9.81 * \sin(40) = 0.530\ Nm \quad \text{................................} (2)$$

Since there will be two motors, the minimum torque required is 0.265Nm. This assumes the robot is going to start moving at the maximum tilt angle, in reality inertia also has to be considered.

Looking at practical example, Gornicki used motors with a stall torque of 0.224Nm and a gear with a 3:1 ratio [11]. Assuming 15% inefficiency [12], that equates to 0.5712Nm.

To fit the requirements, the chosen motor is the Pololu medium power 47:1 Metal Gearmotor with 48 CPR Encoder. The stall torque of the motor is 0.611Nm and the encoder outputs 2248.86 counts per revolution [13], corresponding to a resolution of up to 0.16°. The encoders are necessary for odometry, without the encoders the robot may balance but it will be moving around constantly.

5.3. Power Source

The considered power sources were lithium polymer (Li-Po) batteries and AA batteries. Li-Po batteries were found to be the most appropriate power source, as AA batteries generally have a lower maximum discharge current [14]. Li-Po batteries also have a relatively high specific energy and energy density [15]. There are some dangers associated with them; these have been addressed in the Health and Safety Risk Assessment (Appendix 2). The specific battery to be used is the Turnigy 3 cell 2200mAh 20C. The stall current for each motor is 2.1A at 12V [13] and power also needs to be supplied to the other devices (Arduino, IMU and encoders). As a rough estimate, the power source should be able to supply a minimum of 5A. The Li-Po battery can supply up to 44A [16].

5.4. Motor Driver Board

The L298 dual full bridge driver was the initial choice due to its popularity. According to the datasheet the motor driver has peak output current per channel of 2A in DC operation and up to 3A non-repetitive [17]. In practice, the L298 would go into thermal shut down at 0.8A [18], making it unsuitable for the robot. To avoid deceit from manufacturers, the L6203 was chosen, theoretically it can supply 5A [19]. In order not to damage the motors resettable fuses will be used.

5.5. IMU

The selected IMU is the MPU 9250 by InvenSense. It has 9 degrees of freedom, consisting of 3-axis gyroscope, 3-axis accelerometer and 3-axis magnetometer. The magnetometer is not necessary, but the IMU without the magnetometer costs twice the price. By accessing the configuration register, the gyroscope full scale range can adjusted from ±250 to 1000 degrees per second. The accelerometer can also be programmed from ±2 to 6 g. The device has a built in Digital Motion Processor (DMP), but for this project it will not be used. A great advantage of this IMU is that it has been used with the Arduino and libraries are available for it. [20]

Communication between the Arduino and the IMU is through the Inter-Integrated Circuit (I2C) protocol. To read the values form the gyroscope and accelerometer, specific memory addresses need to be accessed (the register map is in the Appendix 6). Following a tutorial for the MPU6050, the raw data values were read. Surprisingly, the register map for MPU9250 is identical to the MPU6050. The code to read the values and the output window are in the Appendix 7 and 8, respectively**.**

5.6. Overall Design

The overall planned format of the robot can be seen in the Solidworks render below:

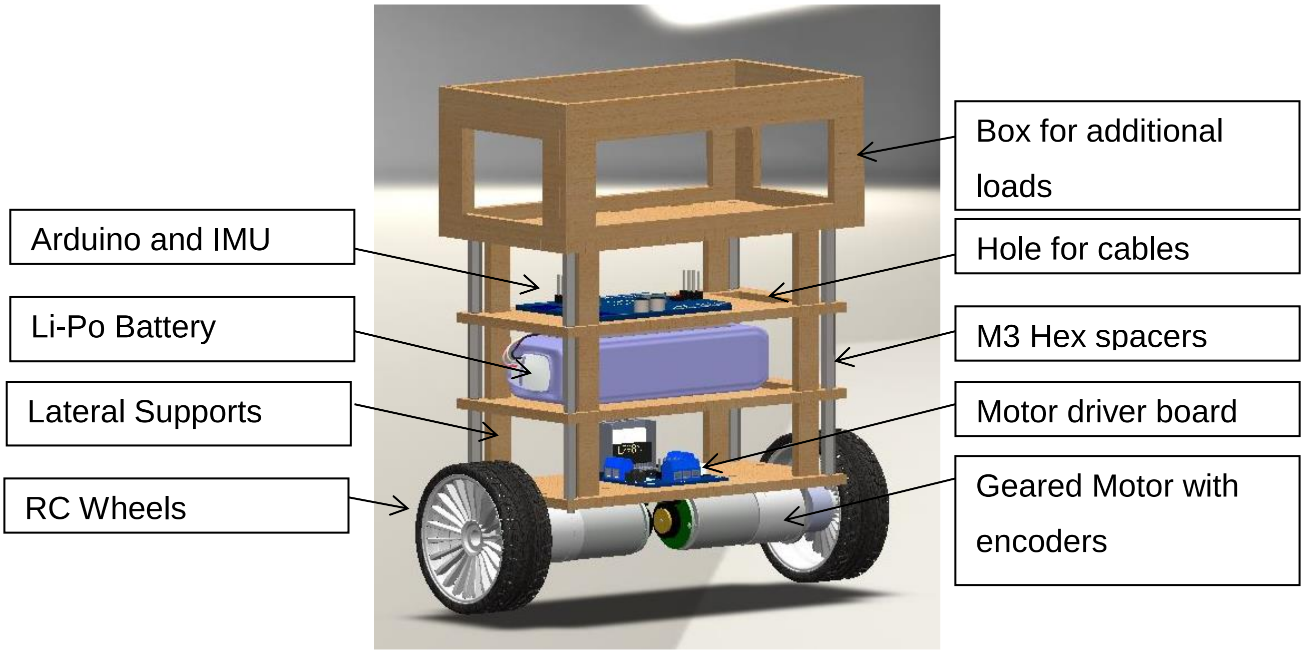

Figure 5 – Solidworks render of Robot and Parts Diagram

The design is an adaptation of the SainSmart self-balancing robot [21]. The design is entirely modular. The layer heights can be adjusted by choosing spacers of different lengths and the box for loads can be removed. Having the layer format also protects the components, specifically the Li-Po battery. The battery is shielded from heat coming from the motor drivers and it is also protected from impacts.

The layers will be made of Medium Density Fibreboard (MDF). It is relatively light, inexpensive, easy to manufacture and readily available in the university. In addition MDF should be able to withstand the drops and hits that might happen when the robot controller is being tuned.

The wheels will be from Remote Controlled (RC) cars. RC wheels are often wide giving a larger surface area in contact with ground and the tyres are made of soft rubber. They are designed this way to have good grip as often RC hobbyists compete with each other. Having good traction is essential or the robot may skid and fall.

6. Conclusion

A basic understanding of Kalman filters has been achieved and the robot's physical design has been completed. The next step this semester is to implement the KF in C code to fuse the data from the gyroscope and accelerometer. A comparison can then be made between the data from the output of the KF and the built in DMP. Once the Kalman filter is well tuned and a good estimate of the tilt angle is obtained, the PID controller can then be developed to maintain the robot upright.

The progress achieved to date is as planned, this suggests that the aim of the project is realistic. Based on the Project Plan in Appendix 3, the project should be completed by the end of week 6 in the second semester. This allows some time to adjust for unpredicted scenarios or to be dedicated in meeting the additional objectives.

7. References

[1] Sundin, C. and Thorstensson, F. (2013). Autonomous balancing robot. Masters of Science. Chalmers University of Technolgy.

[2] Welch, G. and Bishop, G. (2001). An Introduction to the Kalman Filter. [online] Available at: http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf [Accessed 4 Nov. 2016].

[3] Bonafilia, B., Gustafsson, N., Nyman, P. and Nilsson, S. (n.d.). Self-balancing two-wheeled robot. Chalmers University of Technology.

[4] Ding, Y., Gafford, J. and Kunio, M. (2012). Modeling, Simulation and Fabrication of a Balancing Robot. Harvard University, Massachusetts Institute of Technology.

[5] Welch, G. and Bishop, G. (2006). An Introduction to the Kalman Filter. [online] Available at: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf [Accessed 4 Nov. 2016].

[6] Carrasco, J. (2016). 5 Lines of Code to Land on the Moon.

[7] Faragher, R. (2012). Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation. IEEE SIGNAL PROCESSING MAGAZINE, [online] (1053-5888/12), pp.128-132. Available at: https://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf [Accessed 3 Nov. 2016].

[8] Cornman, A. and Mei, D. (n.d.). Extended Kalman Filtering. Stanford University.

[9] Ooi, R. (2013). Balancing a Two-Wheeled Autonomous Robot. Undergraduate. The University of Western Australia.

[10] Esme, B. (2016). Bilgin's Blog | Kalman Filter For Dummies. [online] Bilgin.esme.org. Available at: http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies [Accessed 4 Nov. 2016].

[11] Gornicki, K. (2015). Autonomous Self Stabilising Robot. Undergraduate. The University of Manchester.

[12] Baines, G. (2015) 'Embedded Systems Project: Motor Characterisation and Gearbox Ratio Selection'. Available at: https://online.manchester.ac.uk/bbcswebdav/pid-3643166-dt-content-rid-12476223_1/courses/I3027-EEEN-21000-1151-1YR-

027927/ESP%20Week%202%20Motors%20and%20Gearbox_2015.pdf [Accessed 4 Nov 2015]

[13] Pololu.com. (2016). Pololu - 47:1 Metal Gearmotor 25Dx52L mm MP 12V with 48 CPR Encoder. [online] Available at: https://www.pololu.com/product/3241/specs [Accessed 3 Nov. 2016].

[14] Energizer.com. (2016). Product Datasheet - L91 Ultimate Lithium. [online] Available at: http://data.energizer.com/PDFs/l91.pdf [Accessed 5 Nov. 2016].

[15] Learn.sparkfun.com. (2016). Battery Technologies. [online] Available at: https://learn.sparkfun.com/tutorials/battery-technologies [Accessed 4 Nov. 2016].

[16] Hobbyking. (2016). Turnigy 2200mAh 3S 20C Lipo Pack. [online] Available at: https://www.hobbyking.com/en_us/turnigy-2200mah-3s-20c-lipo-pack.html [Accessed 4 Nov. 2016].

[17] Sparkfun. (2016). L298 H Bridge Datasheet. [online] Available at: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf [Accessed 3 Nov. 2016].

[18] Rugged Circuits. (2016). The Motor Driver Myth. [online] Available at: http://www.rugged-circuits.com/the-motor-driver-myth/ [Accessed 3 Nov. 2016].

[19] Anon, (2016). L6203 Datasheet. [online] Available at: http://users.ece.utexas.edu/~valvano/Datasheets/L6203.pdf [Accessed 4 Nov. 2016].

[20] Invensense.com. (2016). MPU-9250 | InvenSense. [online] Available at: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250 [Accessed 4 Nov. 2016].

[21] Sainsmart.com. (2016). SainSmart 2-Wheel Arduino Self-Balancing Robot Kit 3D Printing, Arduino, Robotics | Sainsmart. [online] Available at: http://www.sainsmart.com/sainsmart-balancing-robot-kit.html [Accessed 3 Nov. 2016].

[22] van Biezen, M. (2015). The Kalman Filter (2 of 55). [online] Ilectureonline.com. Available at: http://www.ilectureonline.com/lectures/subject/SPECIAL%20TOPICS/26/190/1963 [Accessed 6 Nov. 2016].

8. Appendices

8.1. Appendix 1 –Technical Risk Assessment

As mentioned previously, the Arduino makes it an easy platform to program in due to large community and extensive collection of libraries. Furthermore, Kalman filters and balancing robots have been realized using an Arduino, this suggests a lower technical risk. However, due to low processing capability the Arduino itself may be a liability. Christian Sundin mentions that the Arduino could not execute the algorithm for an LQG controller fast enough [1]. If met with such scenario, a solution may be to use two Arduinos in a master-slave configuration or use a faster microcontroller such as the STM32 Nucleo.

Another risk for the project would be slow order processing time and delivery. If the required components do not arrive within the expected time frame, the project will have to be put on hold. To minimize this risk, component orders were placed early this semester.

8.2. Appendix 2 – Health and Safety Risk Assessment

WORK ACTIVITY/ WORKPLACE (WHAT PART OF THE ACTIVITY POSES RISK OF INJURY OR ILLNESS)	HAZARD (S) (SOMETHING THAT COULD CAUSE HARM, ILLNESS OR INJURY)	LIKELY CONSEQUENCES (WHAT WOULD BE THE RESULT OF THE HAZARD)	WHO OR WHAT IS AT RISK (INCLUDE NUMBERS AND GROUPS)	EXISTING CONTROL MEASURES IN USE (WHAT PROTECTS PEOPLE FROM THESE HAZARDS)	WITH EXISTING CONTROLS				MEASURE REQUIRED TO PREVENT OR REDUCE RISK (WHAT NEEDS TO BE DONE TO MAKE THE ACTIVITY AS SAFE AS POSSIBLE)	PERSON RESPONSIBLE FOR ACTIONS AND AGREED TIMESCALES TO ACHIEVE THEM	WITH NEW CONTROLS			
					SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE			SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE
Soldering	High Temperature of the Soldering iron	Burns	Abdul Gafar and people nearby	Holder for iron, sponge to test if hot	2	2	4	Y	Risks considered acceptable	Abdul Gafar 2016 - 2017 academic year	2	2	4	Y
	Inhaling solder fumes	Illness		Fume extractor	3	1	3	Y	Risks considered acceptable	Abdul Gafar 2016 - 2017 academic year	3	1	3	Y
Programming for the robot	Using a computer	RSI	Abdul Gafar	Ergonomic chairs	1	3	3	Y	Risks considered acceptable	Abdul Gafar 2016 - 2017 academic year	1 1	3 3	3 3	Y Y
		Eye strain		Limited computer usage	1	3	3	Y						
Wiring the circuit	Using wire cutters	Cutting hands	Abdul Gafar	n/a	2	2	4	Y	Risks considered acceptable	Abdul Gafar 2016 - 2017 academic year	2 3	2 1	4 3	Y Y
		Small bits of wire flying off and hitting the eye			3	1	3	Y						
Using / Charging a Lithium Polymer Battery	Battery could explode if: overcharged, heated, the output terminals are shorted and/or is punctured	Damage equipment around and/or cause burns	Equipment in proximity and/or people nearby	Use a Balance charger, the battery will be shielded from heat or impact, a fuse should be used	4	2	8	Y	Ensure battery is charged under supervision	Abdul Gafar 2016 - 2017 academic year	4	1	4	Y
	Damage the rest of the circuitry, due to high current	Some circuitry of the robot could be damaged	Robot gets damaged	Fuse	2	2	4	Y	Risks considered acceptable	Abdul Gafar 2016 - 2017 academic year	2	2	4	Y

Assessment ID Number (E&EE_AG_09/10/2016_9097951).. **Activity Location: SSB C34**..

				THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (MAX 12 MTHS)
MANAGER/SUPERVISOR	NAME: Dr. Joaquin Carrasco Gomez	SIGNED:	DATE:	
Student:	NAME: Abdul Gafar	SIGNED:	DATE:	

IF THE ANSWERS TO ANY OF THE QUESTIONS BELOW IS YES THEN ADDITIONAL SPECIFIC RISK ASSESSMENTS MAY BE REQUIRED.

IS THERE A RISK OF FIRE?	Y/N	DOES THE ACTIVITY REQUIRE ANY HOME WORKING?	Y/N
ARE SUBSTANCES THAT ARE HAZARDOUS TO HEALTH USED?	Y/N	ARE THE EMPLOYEES REQUIRED TO WORK ALONE	Y/N
IS THERE MANUAL HANDLING INVOLVED?	Y/N	DOES THE ACTIVITY INVOLVE DRIVING	Y/N
IS PPE WORN OR REQUIRED TO BE WORN?	Y/N	DOES THE ACTIVITY REQUIRE WORK AT HEIGHT	Y/N
ARE DISPLAY SCREENS USED?	Y/N	DOES THE ACTIVITY INVOLVE FOREIGN TRAVEL	Y/N
IS THERE A SIGNIFICANT RISK TO YOUNG PERSONS?	Y/N	IS THERE A SIGNIFICANT RISK TO NEW / PREGNANT MOTHERS?	Y/N

Severity value **= potential consequence of an incident/injury**

5	Very High	Death / permanent incapacity / widespread loss
4	High	Major Injury (Reportable Category) / Severe Incapacity / Serious Loss
3	Moderate	Injury / illness of 3 days or more absence (reportable category) / Moderate loss
2	Slight	Minor injury / illness – immediate First Aid only / slight loss
1	Negligible	No injury or trivial injury / illness / loss

Likelihood value **= what is the potential of an incident or injury occurring**

5	Almost certain to occur
4	Likely to occur
3	Quite possible to occur
2	Possible in current situation
1	Not likely to occur

risk rating = severity value × likelihood value

risk ratings **are classified as *low* (1 – 5), *medium* (6 – 9) and *high* (10 – 25)**

Risk Classification and Actions:

Rating	Classification	Action
1 – 5	Low	Tolerable risk - Monitor and Manage
6 – 9	Medium	Review and introduce additional controls to mitigate to "As Low As Reasonably Practicable" (ALARP)
10 – 25	High	Stop work immediately and introduce further control measures

LIKELIHOOD \ SEVERITY	1	2	3	4	5
1	Low	Low	Low	Low	Low
2	Low	Low	Medium	Medium	High
3	Low	Medium	Medium	High	High
4	Low	Medium	High	High	High
5	Low	High	High	High	High

8.3. Appendix 3– Project Plan

SEMESTER 1													
TASKS	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Progress Report Deadline (07/11)	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Research Kalman Filters												Contingency, in case of delays in delivery of components or certain tasks take longer than expected	
Implement Basic Kalman Filter in MATLAB													
Design Robot													
Order Components													
Establish I2C comm. with IMU													
Draft Progress Report													
Finish Progress Report													
Write the Kalman Filter code for the Arduino													
Design Motor Breakout PCB													
Assemble Robot													

SEMESTER 2														
TASKS	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	EASTER BREAK (31/03 - 23/04)	Week 10	Poster and Final Report Deadline (01/05)	Week 11	Week 12
Write code for encoders (calculate displacement)							Contingency, in case certain tasks take longer than expected or alterations necessary						Oral Examination and Poster	
Tune the Kalman Filter														
Develop PID controller														
Combine code for Filter, encoders and controller														
Testing / further tuning														
Write section for final report based on progress														
Design Poster / Prepare for Viva Voce														
Compile / Draft Report														

8.4. Appendix 4 -Kalman Filter Code 1 – Constant

```
%Example from http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies
zk = zeros (1,200);
y = 4 * ones (1,200);

for n=1:200
    zk (n) = 4 + 0.5*randn;
end

x0=0;
P0=1;
R=0.25;
A=1;
Q=0;

x = zeros (1,200);
k = zeros (1,200);
p = zeros (1,200);

k(1)= P0/(P0+R);

x(1)= x0 + k(1)*(zk(1)-x0);
p(1)= (1-k(1))*P0;


for t=2:200

   k(t)= p(t-1)/(p(t-1)+R);
   x(t)= x(t-1) + k(t)*(zk(t)-x(t-1));
   p(t)=(1-k(t))*p(t-1);

end
subplot(121)
plot(x)
hold on
plot (y, 'Color','r')

subplot(122)
plot(x-4)
```

8.5. Appendix 5 - Kalman Filter Code 2 – Linear

```
%An object travelling in 1D at a constant velocity of 1.5m/s

yk = zeros (1,200);
for n=1:200yk
    yk (n) = 1.5*n + 3*randn ;
end  % creates 'measured' inputs with 'measurements' being independent
       %of each other i.e. erros don't propagate

R=1; %the function 'randn' ouputs normally distributed random numbers
     %this makes the standard deviation=1, therefore variance=1

X0=0; %starting at origin
P0=1; %any non-zero value otherwise K=0
A=1;
Q=0;
U=1.5; %travelling speed
W=0;   %Assuming no white noise
H=1; %1 as just numbers not matrices

B = zeros (1,200);
for n=1:200
    B(n)= n;
end    %for elapsed time

xkp = zeros (1,200);
x = zeros (1,200);
k = zeros (1,200);
pkp = zeros (1,200);
pk = zeros (1,200);

%t1 Predicted state
xkp(1)= A*X0 +  B(1)*U + W;
pkp(1)= A*P0*A + Q;

%update w/ new measurements and kalman gain
k(1)=(pkp(1)*H)*inv(H*pkp(1)*H + R);
x(1)= xkp(1)  + k(1)*(yk(1)-H*xkp(1));
pk(1)= (1-k(1)*H)*pkp(1);

for t=2:200
    %t(n) Predicted state
    xkp(t)= A*x(t-1)  +  1*U + W;
    pkp(t)= A*pk(t-1)*A + Q;
    %update w/ new measurements and kalman gain
    k(t)=(pkp(t)*H)*inv(H*pkp(t)*H + R);
    x(t)= xkp(t)  + k(t)*(yk(t)-H*xkp(t));

    pk(t)= (1-k(t-1)*H)*pkp(t-1);
end

test = linspace(0,300,200);

subplot(121)
plot(x)
hold on
plot (yk, 'Color','r')
subplot(122)
plot(x-test)
hold on
plot(yk-test, 'Color','r')
```

8.6. Appendix 6 - MPU 9250 Register Map

InvenSense	MPU-9250 Register Map and Descriptions	Document Number: RM-MPU-9250A-00 Revision: 1.4 Release Date: 9/9/2013

3 Register Map for Gyroscope and Accelerometer

The following table lists the register map for the gyroscope and accelerometer in the MPU-9250 MotionTracking device.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00	0	SELF_TEST_X_GYRO	R/W	xg_st_data [7:0]							
01	1	SELF_TEST_Y_GYRO	R/W	yg_st_data [7:0]							
02	2	SELF_TEST_Z_GYRO	R/W	zg_st_data [7:0]							
0D	13	SELF_TEST_X_ACCEL	R/W	XA_ST_DATA [7:0]							
0E	14	SELF_TEST_Y_ACCEL	R/W	YA_ST_DATA [7:0]							
0F	15	SELF_TEST_Z_ACCEL	R/W	ZA_ST_DATA [7:0]							
13	19	XG_OFFSET_H	R/W	X_OFFS_USR [15:8]							
14	20	XG_OFFSET_L	R/W	X_OFFS_USR [7:0]							
15	21	YG_OFFSET_H	R/W	Y_OFFS_USR [15:8]							
16	22	YG_OFFSET_L	R/W	Y_OFFS_USR [7:0]							
17	23	ZG_OFFSET_H	R/W	Z_OFFS_USR [15:8]							
18	24	ZG_OFFSET_L	R/W	Z_OFFS_USR [7:0]							
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	FIFO_MODE	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	XGYRO_Ct en	YGYRO_Ct en	ZGYRO_Ct en	GYRO_FS_SEL [1:0]		-	FCHOICE_B[1:0]	
1C	28	ACCEL_CONFIG	R/W	ax_st_en	ay_st_en	az_st_en	ACCEL_FS_SEL[1:0]		-		
1D	29	ACCEL_CONFIG 2	R/W	-				ACCEL_FCHOICE_B		A_DLPF_CFG	
1E	30	LP_ACCEL_ODR	R/W	-				Lposc_clksel [3:0]			
1F	31	WOM_THR	R/W	WOM_Threshold [7:0]							
23	35	FIFO_EN	R/W	TEMP_FIFO_EN	GYRO_XOUT	GYRO_YOUT	GYRO_ZOUT	ACCEL	SLV2	SLV1	SLV0
24	36	I2C_MST_CTRL	R/W	MULT_MST_EN	WAIT_FOR_ES	SLV_3_FIFO_EN	I2C_MST_P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0_RNW	I2C_ID_0 [6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0_EN	I2C_SLV0_BYTE_SW	I2C_SLV0_REG_DIS	I2C_SLV0_GRP	I2C_SLV0_LENG[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1_RNW	I2C_ID_1 [6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1_EN	I2C_SLV1_BYTE_SW	I2C_SLV1_REG_DIS	I2C_SLV1_GRP	I2C_SLV1_LENG[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2_RNW	I2C_ID_2 [6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2_EN	I2C_SLV2_BYTE_SW	I2C_SLV2_REG_DIS	I2C_SLV2_GRP	I2C_SLV2_LENG[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3_RNW	I2C_ID_3 [6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3_EN	I2C_SLV3_BYTE_SW	I2C_SLV3_REG_DIS	I2C_SLV3_GRP	I2C_SLV3_LENG [3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4_RNW	I2C_ID_4 [6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4_EN	SLV4_DONE_INT_EN	I2C_SLV4_REG_DIS	I2C_MST_DLY[4:0]				

7 of 55

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_THROUGH	I2C_SLV4_DONE	I2C_LOST_ARB	I2C_SLV4_NACK	I2C_SLV3_NACK	I2C_SLV2_NACK	I2C_SLV1_NACK	I2C_SLV0_NACK
37	55	INT_PIN_CFG	R/W	ACTL	OPEN	LATCH_INT_EN	INT_ANYRD_2CLEAR	ACTL_FSYNC	FSYNC_INT_MODE_EN	BYPASS_EN	-
38	56	INT_ENABLE	R/W	-	WOM_EN	-	FIFO_OFLOW_EN	FSYNC_INT_EN	-	-	RAW_RDY_EN
3A	58	INT_STATUS	R	-	WOM_INT	-	FIFO_OFLOW_INT	FSYNC_INT	-	-	RAW_DATA_RDY_INT
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT_H[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT_L[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT_H[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT_L[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT_H[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT_L[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT_H[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT_L[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT_H[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT_L[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT_H[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT_L[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT_H[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT_L[7:0]							
49	73	EXT_SENS_DATA_00	R	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01	R	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02	R	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03	R	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04	R	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05	R	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06	R	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07	R	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08	R	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09	R	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10	R	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11	R	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12	R	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13	R	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14	R	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15	R	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16	R	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17	R	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18	R	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19	R	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20	R	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21	R	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22	R	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23	R	EXT_SENS_DATA_23[7:0]							
63	99	I2C_SLV0_DO	R/W	I2C_SLV0_DO[7:0]							
64	100	I2C_SLV1_DO	R/W	I2C_SLV1_DO[7:0]							
65	101	I2C_SLV2_DO	R/W	I2C_SLV2_DO[7:0]							

InvenSense	MPU-9250 Register Map and Descriptions	Document Number: RM-MPU-9250A-00 Revision: 1.4 Release Date: 9/9/2013

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
66	102	I2C_SLV3_DO	R/W	I2C_SLV3_DO[7:0]							
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES _SHADOW	-	-	I2C_SLV4 _DLY_EN	I2C_SLV3 _DLY_EN	I2C_SLV2 _DLY_EN	I2C_SLV1 _DLY_EN	I2C_SLV0 _DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO _RST	ACCEL _RST	TEMP _RST
69	105	MOT_DETECT_CTRL	R/W	ACCEL_INT EL_EN	ACCEL_INT EL_MODE	-		-		-	
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST _EN	I2C_IF _DIS	-	FIFO _RST	I2C_MST _RST	SIG_COND _RST
6B	107	PWR_MGMT_1	R/W	H_RESET	SLEEP	CYCLE	GYRO_ STANDBY	PD_PTAT	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	-		DIS_XA	DIS_YA	DIS_ZA	DIS_XG	DIS_YG	DIS_ZG
72	114	FIFO_COUNTH	R/W	-			FIFO_CNT[12:8]				
73	115	FIFO_COUNTL	R/W	FIFO_CNT[7:0]							
74	116	FIFO_R_W	R/W	D[7:0]							
75	117	WHO_AM_I	R	WHOAMI[7:0]							
77	119	XA_OFFSET_H	R/W	XA_OFFS [14:7]							
78	120	XA_OFFSET_L	R/W	XA_OFFS [6:0]							-
7A	122	YA_OFFSET_H	R/W	YA_OFFS [14:7]							
7B	123	YA_OFFSET_L	R/W	YA_OFFS [6:0]							-
7D	125	ZA_OFFSET_H	R/W	ZA_OFFS [14:7]							
7E	126	ZA_OFFSET_L	R/W	ZA_OFFS [6:0]							-

Table 1 MPU-9250 mode register map for Gyroscope and Accelerometer

Note: Register Names ending in _H and _L contain the high and low bytes, respectively, of an internal register value.

In the detailed register tables that follow, register names are in capital letters, while register values are in capital letters and italicized. For example, the ACCEL_XOUT_H register (Register 59) contains the 8 most significant bits, *ACCEL_XOUT*[15:8], of the 16-bit X-Axis accelerometer measurement, *ACCEL_XOUT*.

The reset value is 0x00 for all registers other than the registers below.

- Register 107 (0x01) Power Management 1
- Register 117 (0x71) WHO_AM_I

9 of 55

8.7. Appendix 7 - IMU Code to obtain raw values

```
// code modified from https://www.youtube.com/watch?v=M9lZ5Qy5S2s
#include <Wire.h>
long accelX, accelY, accelZ; //accelerometer
long gyroX, gyroY, gyroZ;//gyro

void setup() {
  Serial.begin(9600);
  Wire.begin(); // starting I2C communication

  // initialising the sensor  //SETTING UP POWER
  Wire.beginTransmission(0x68); //I2C address of the MPU (as SJ2 is in place)
  Wire.write(0x6B); // Power Management 1
  Wire.write(0x00); // pg 40
  Wire.endTransmission();

  Wire.beginTransmission(0x68); //I2C address of the MPU (as SJ2 is in place)
  Wire.write(0x6C); // Power Management 2
  Wire.write(0x00); // pg 41 - enables gyro and acc x,y,z
  Wire.endTransmission();

  //GYRO CONFIGURATION
  Wire.beginTransmission(0x68); //I2C address of the MPU (as SJ2 is in place)
  Wire.write(0x1B); // gyro configuration
  Wire.write(0x00); // pg 14 - sets the full scale to +/- 250 degress/second
  Wire.endTransmission();

  //ACC CONFIGURATION
  Wire.beginTransmission(0x68); //I2C address of the MPU (as SJ2 is in place)
  Wire.write(0x1C); // acc configuration
  Wire.write(0x00); // pg 14 - sets the full scale to +/- 2gs
  Wire.endTransmission();
}

void loop() {
//get raw data (does not represent gs or dps, needs to be scaled depending on setup)
```

```
//accelerometer readings
  Wire.beginTransmission(0x68); //I2C address of the MPU
  Wire.write(0x3B); //Starting register for Accel Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); //Request Accel Registers (3B - 40)
  while(Wire.available() < 6);
  accelX = Wire.read()<<8|Wire.read(); //Store first two bytes into accelX
  accelY = Wire.read()<<8|Wire.read(); //Store middle two bytes into accelY
  accelZ = Wire.read()<<8|Wire.read(); //Store last two bytes into accelZ

//gyro data
  Wire.beginTransmission(0x68); //I2C address of the MPU
  Wire.write(0x43); //Starting register for Gyro Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); //Request Gyro Registers (43 - 48)
  while(Wire.available() < 6);
  gyroX = Wire.read()<<8|Wire.read(); //Store first two bytes into accelX
  gyroY = Wire.read()<<8|Wire.read(); //Store middle two bytes into accelY
  gyroZ = Wire.read()<<8|Wire.read(); //Store last two bytes into accelZ

  Serial.print("Gyro");
  Serial.print(" X=");
  Serial.print(gyroX);
  Serial.print(" Y=");
  Serial.print(gyroY);
  Serial.print(" Z=");
  Serial.print(gyroZ);
  Serial.print(" Accel");
  Serial.print(" X=");
  Serial.print(accelX);
  Serial.print(" Y=");
  Serial.print(accelY);
  Serial.print(" Z=");
  Serial.println(accelZ);
}
```

8.8. Appendix 8 – IMU Output