Stabilisation of a Pan-and-Tilt Unit holding a camera

Final Report for CS39440 Major Project

Author: Edgar Ivanov (edi@aber.ac.uk)
Supervisor: Fred Labrosse (ffl@aber.ac.uk)

May 1, 2014
Version: -999999999.045879 (Draft of the Draft of the Draft)

This report was submitted as partial fulfilment of a BSc degree in Mobile And Wearable Computing (G421)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature ....

Date ....

Abstract

Computer Science department at Aberystwyth University has a large all-terrain rover that is equipped with a panoramic camera mounted on a Pan-and-Tilt Unit (PTU) that is stabilised using gyroscopes. One of the problems of the gyroscopes is that they tend to drift and therefore need to be regularly corrected to ensure the camera stays vertical. The project is about building a small hardware module and the software to drive it that will interface tilt sensor (inclinometer) with the PTU to ensure camera ”verticality”.

CONTENTS

Chapter 1

Introduction

1.1 Overview

Pan-and-Tilt Unit (PTU) is a stabilised 2-axes platform mounted on the Idris electric vehicle that holds a panoramic camera. This vehicle is used for the research and drives over the curved surfaces [5]. PTU has a built in functionality to stabilize at the chosen position: in case the rover drives up the hill camera will stay vertical. In order to achieve camera verticality stabilization command could only be invoked once, when the rover is standing on the flat surface. Its position could also be adjusted manually if it is on a curved surface and then stabilization command could be issued.

1.2 Problem analysis

To ensure stability of the camera PTU uses gyroscopes, which would be enough on its own in a perfect world. However, the well known problem of the gyroscopes is that their readings are affected by the different air temperature, magnetic effects, friction etc. [7]. When PTU is issued with the stabilization command the platform slowly drifts on both axes and goes to the position which can be up to 20 degrees different from where it should be. This project will concentrate on fixing this particular issue as well as trying to enhance the overall system functionality.

1.3 Aim

The aim of this project is to modify system which is currently in place and implement functionality to stabilize Pan-Tilt Unit with drift compensation. The system should be fully automated and perform drift rate calculation and following calibration on its own, when the appropriate request comes in from the client.

1.4 Proposed Solution

One of the possible solutions could be the use of the additional PTU functionality. It allows to cancel drift by specifying the calculated in advance drift rate in radians per second. The problem with this solution is that the drift rate can change if the surround environment changes (sun goes behind the clouds and air temperature drops affecting gyroscope readings). It is impossible to calculate new drift rate manually every 10 minutes.

The proposed solution is to automate the drift rate calculation so that it can be adjusted while driving. However this approach requires additional information about the current vehicle orientation in the space (its inclination angles with respect to gravity) to be able to predict required PTU position which will then be compared with the actual position and the drift rate that has been calculated. Electronic inclinometer can be used for this purpose. Inclinometer response is an electric signal representing an angle between the internal axis and the gravity vector [7]. Accurate readings can only be obtained using the device if it is in a stable position and does not accelerate, otherwise it will provide erroneous data representing the sum of two vectors earth gravity and acceleration. This imposes certain limitations. PTU drift rate calibration can only be performed when the vehicle does not move.

1.5 Objectives

The aims of this project are:

- Port current system from the Gumstix on to the Raspberry Pi.
- Incorporate new commands for the stabilization and drift rate.
- Implement PTU drift rate calculation functionality.
- Extend client side API to allow new functionality.

1.6 Schedule and Project Management

Due to the tight time-frame imposed for the development of the system, an agile system development methodology will be followed. Agile development has a focus on generating working software prototypes quickly in cycles, rather than the classical approach of thorough documentation and following a strict plan. During software development there are likely to be unexpected technical difficulties that appear and need to be overcome, which can put the project behind the schedule.

The agile methodology attempts to provide the project with a clear direction by regularly assessing the development of the project at the end of each iteration, and interacting with the client to ensure the project is coming closer by fulfilling its requirements. When the progress of the project is assessed regularly, it gives the opportunity to steer it back into the right direction if there are evidences that the course development is moving away from the target. The waterfall method which has a single fixed completion date at which the project completes and should be ready to deployed runs the risk of delays if the system does not come together as planned. If the project is abandoned at any point before it is completed there is unlikely to be even a partially working system completed. Thus the agile approach greatly reduces both development time and costs [9].

Following the initial evaluation of the different software development methodologies the feature-driven development methodology was chosen. At different stages of this project work will focus on the different parts of the system. In each such part there is a set of clearly identifiable features which are required for the final system to work. At the begging of the project consultation with the client (project supervisor) were conducted to identify overall model of the system. At the following meetings overall model was shaped and feature list build. The development of the system will be organised into iterations of relatively short 2 week cycles, with a set of working features produced at the end of each cycle.

Iteration 0 will focus on porting code to the new platform. It is not expected that system will work straight away, but after minor code adjustments it should have basic features working.

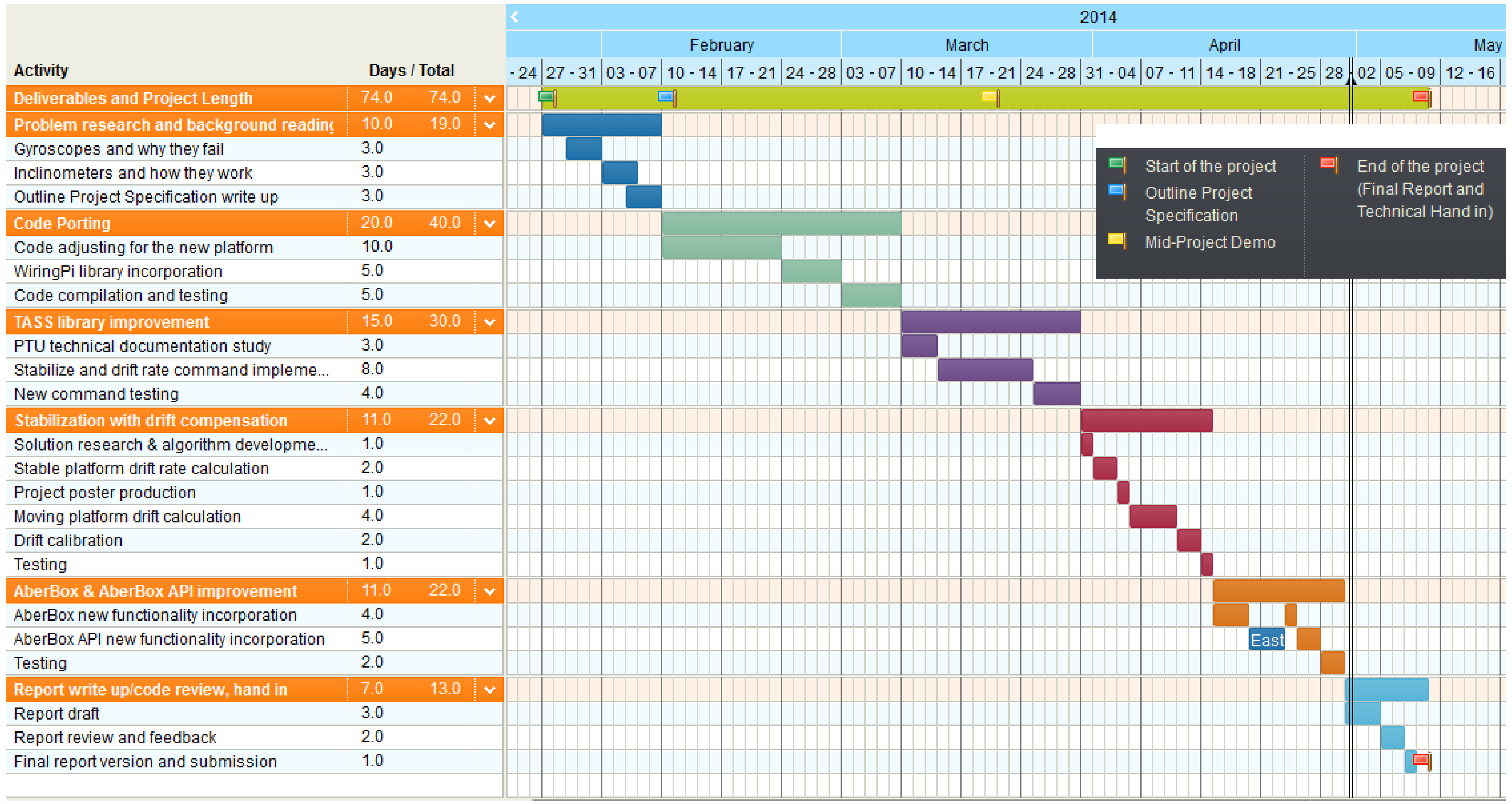
Iteration 1 has an aim of implementing new commands in the PTU TASS library.

Iteration 2 will focus on drift rate calculation algorithm development and testing. This iteration builds on the functionality laid town in the previous cycle and incorporates the ability to calculate drift rate. Following platform stabilization with the calculated drift rate.

Iteration 3 will depend on the work done during all previous cycles. Without successful completion of all previous steps this iteration will not make any sense since there will be nothing to incorporate. In this iteration server side software will be updated to make use of the new features in the PTU TASS library. Client side API will be changed as well to reflect new functionality of the server.

At the end of which cycle, there will be a break and client consulted as to whether he feels that the project is continuing in the right direction.

The project schedule on the figure 1.1 illustrates all the main tasks and planed deliverables.



Activity	Days	Total
Deliverables and Project Length	74.0	74.0
Problem research and background readinç	10.0	19.0
Gyroscopes and why they fail	3.0	
Inclinometers and how they work	3.0	
Outline Project Specification write up	3.0	
Code Porting	20.0	40.0
Code adjusting for the new platform	10.0	
WiringPi library incorporation	5.0	
Code compilation and testing	5.0	
TASS library improvement	15.0	30.0
PTU technical documentation study	3.0	
Stabilize and drift rate command impleme...	8.0	
New command testing	4.0	
Stabilization with drift compensation	11.0	22.0
Solution research & algorithm developme...	1.0	
Stable platform drift rate calculation	2.0	
Project poster production	1.0	
Moving platform drift calculation	4.0	
Drift calibration	2.0	
Testing	1.0	
AberBox & AberBox API improvement	11.0	22.0
AberBox new functionality incorporation	4.0	
AberBox API new functionality incorporation	5.0	
Testing	2.0	
Report write up/code review, hand in	7.0	13.0
Report draft	3.0	
Report review and feedback	2.0	
Final report version and submission	1.0	

Figure 1.1: Project schedule

Chapter 2

Design

2.1 Control System

This project will be based on the system that is currently used to control PTU and takes readings from the inclinometer. Current system consists of the Gumstix minicomputer, gpio14 chip, relays, inclinometer, PTU, server side code (running on Gumstix), client side code (provides API for the interaction with the main system) and the library implementing PTU TASS communication protocol to interact with the PTU.

There was a decision made by the client to replace the platform (Gumstix) that is currently used. The rationale behind the decision were client's concerns about the currently used Gumstix computer which is getting old. In case of a breakdown it would be difficult to find the replacement parts. One of the features requested was the ability to compile accompanying code on the platform itself (this is currently impossible on the Gumstix due to the hardware limitations) instead of cross-compiling code on the PC and then uploading executables to the Gumstix. Following a thorough discussion Raspberry Pi minicomputer was chosen as a replacement platform. It has enough power to compile the code as well as all the required interfaces that can be connected to the other equipment. PTU TASS library will be reused and its functionality extended to implement the new features. Server side and client side code will be adjusted to provide access to the new functionality.

2.2 System Architecture Overview

As discussed in the 2.1 section, there will be both hardware and software parts in this project, they will be discussed further in this chapter. The server side will consist from the software running on the Raspberry PI minicomputer and handling all client requests. Connection will be initiated by the client over the TCP/IP protocol. Client side application will be using provided AberBox API to send/receive commands. Overall system architecture is presented on the figure 2.1

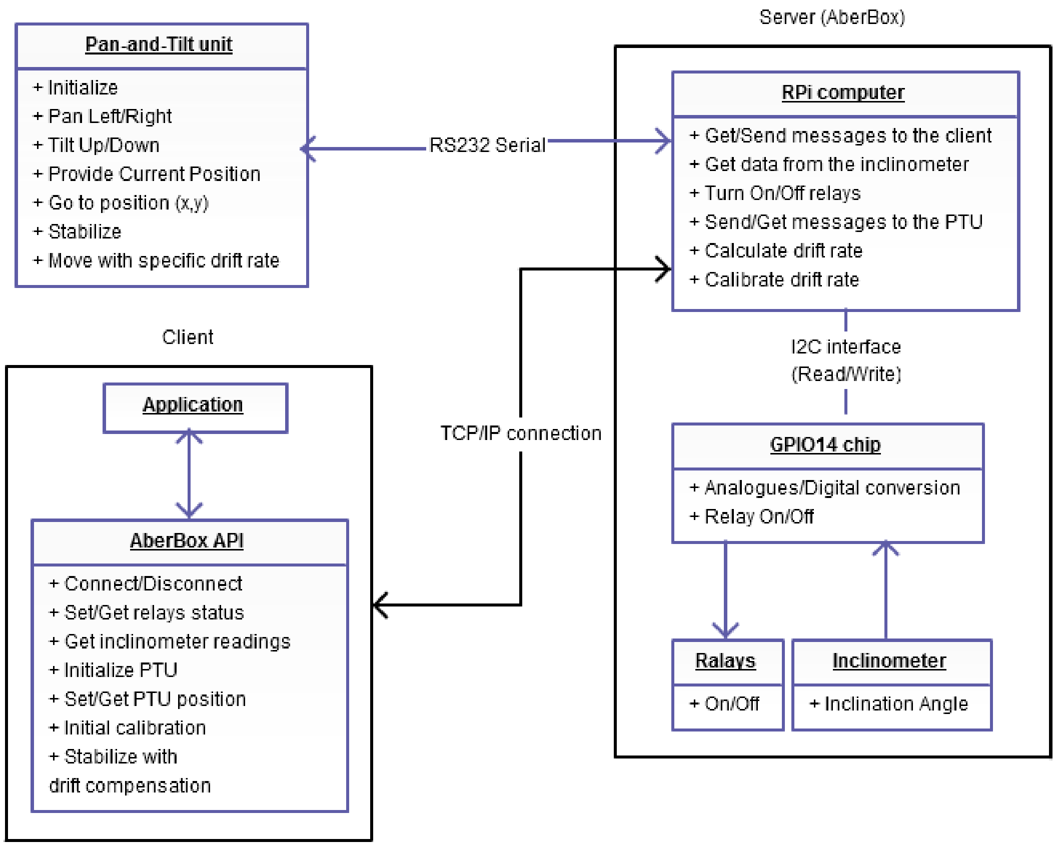


Figure 2.1: System Architecture Overview

2.3 Hardware Design

The proposed system will consist of the Raspberry Pi minicomputer(RPi), analogue inclinometer, gpio14 chip, relays and the PTU.

2.3.1 RPi

As a platform for the main control system Raspberry Pi minicomputer was chosen. It will replace the currently employed for this task Gumstix single board computer which provides limited control over the PTU.

RPi is a credit-card-sized single-board computer with a 512 MiB of RAM and 700 MHz ARM based CPU. It is powerful enough for the proposed tasks to be completed and have all the required interfaces to be connected to the other peripheral. It has GPIO pins, including SPI and I^{2}C interfaces, UART serial console, 5v and GND supply pins. Such a powerful device may be an overkill for this task, but the decision was made by the client.

2.3.2 Inclinometer

An inclinometer will be used to get data about the current chassis position in the space. The client suggested to use the SCA121T dual axis inclinometer (figure 2.2) bolted to the chassis of the rover.

It will provide the information about the inclination angles during calibration. We will need a A/D converter, since it is a analogue device.

Figure 2.2: SCA121T-D05 Inclinometer

2.3.3 GPIO14 chip

The GPIO14 chip is a pre-programmed PIC16F818 micro controller. It intends to provide general purpose I/O expansion on the I^2C bus. It has 14 general purpose I/O lines and 5 analogues input channels with 10-bit A/D conversion [2]. GPIO14 chip will be used to convert inclinometer signal from the analogue format to digital and to switch on/off relays. This chip will be connected over the I^{2}C bus to the RPi.

2.3.4 PTU

Pan-and-Tilt Unit is a high-precision integrated motion control systems produced by the Sagebrush Technology (now part of the RIEtech Global, LLC [4]). It is designed for the ¡10Kg payloads and is often used to hold cameras, antennas or for instrument positioning. As a part of this project it is used to hold a panoramic camera.

2.4 Software Design

This study will be based on the code that is already used on the Gumstix. It is currently used to send control commands to the PTU, get readings from the inclinometer and switch on/off relays. The code base consists of three main parts: the client side software, the server side software (running on the Gumstix and responding to client calls) and the library that implements TASS protocol.

2.4.1 TASS library

TASS library implements the protocol to communicate with the PTU. All communication is done over the RS-232 interface. To connect it to the RPi (which has TTL interface) RS-232-TTL converter was used. The library implements basic commands, including pan left/right, tilt up/down, get/set position, get/set rotation limits.

One of the objectives of this project is to add implementation for the ’stabilize’ and ’drift rate’ commands as well as logic for the drift rate calculation.

2.4.2 Server Side

The server side software is responsible for the overall control of the PTU and inclinometer. It will be running on Raspberry Pi minicomputer and will respond to the client commands. On of the main challenges is to port the current system from the Gumstix to the RPi platform and make it work.

2.4.3 Clint Side

The client side software provides an API to interact with the server Side. The connection is made over the TCP/IP protocol. The plan is to extend the API upon the creation of the new stabilization functionality.

2.4.4 Operating System

Raspbian is a free operating system that will be running on the RPi. It requires some initial configuration before the main software can be successfully run. Firstly, it requires configuration to prevent Linux from using the serial port; it also needs WiringPi library [6] installed to successfully communicate over the I^2C bus with the GPIO14 chip. All the necessary configuration is covered in Appendix A.

2.5 Overall Architecture

2.6 Some detailed design

2.6.1 Even more detail

2.7 User Interface

2.8 Other relevant sections

Chapter 3

Implementation

Chapter 4

Testing

4.1 PTU TASS library testing

ID	Requirement	Description	Input	Expected output	Pass/ Fail	Comment
1	FR1	Test PTU connection and initialization	Connect to PTU and perform fast initialization	PTU platform should rotate to identify its limits	P	
2	FR2	Send tilt Up/Down commands to the PTU	”TU/TD” commands are sent	PTU platform should tilt UP/Down	P	
3	FR3	Send pan Left/Right commands to the PTU	”PL/PR” commands are sent	PTU platform should pan UP/Down	P	
4	FR4	Keep platform stabilized.	”HI” command is sent to the PTU.	PTU should adjust platform position to keep it vertical when whole unit is tilted.	P	Platform position changes when whole unit is tilted.
5	FR5	Send drift rate to the PTU	”*mrf,f” command is sent.	Platform is stabilized with the given drift rate	P	
6	FR6		Platform is stabilized with the given drift rate	P		

Chapter 5

Evaluation

Appendices

Appendix A

Libraries and RPi configuration

This Appendix contains essential information on the RPi configuration. This configuration is required for the proper server side software functionality.

1.1 Raspbian OS configuration

The installation procedure of the Raspbian OS is covered on this website: http://www.raspbian.org/

1.2 Preventing Linux using the serial port

Disabling the serial console is required to use the Raspberry Pi's serial port (UART) to talk to other devices. There is a script which allows to automate the whole process and is covered in detail on this page: https://github.com/lurch/rpi-serial-console .

1.3 WiringPi library

The WiringPi project has been used to read and write data on I^2C bus as part of the interaction with the gpio14 chip. The library is open source and it is available from the Gordons Projects website: http://wiringpi.com/ . Installation is covered on this web page: http://wiringpi.com/download-and-install/ . Before the I^2C interface can be used I^2C drivers need to be load into the kernel, guide is available at the http://wiringpi.com/reference/i2c-library/ .

Appendix B

Code samples

2.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*---------------------------------------------------*/
  /* Minimum Standard Random Number Generator          */
  /* Taken from Numerical recipies in C                */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity     */
  /* Always call with negative number to initialise    */
  /*---------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
```

```
double temp;

if (*idum <=0)
{
  if (-(*idum) < 1)
  {
    *idum = 1;
  }else
  {
    *idum = -(*idum);
  }
  idum2=(*idum);
  for (j=NTAB+7;j>=0;j--)
  {
    k = (*idum)/IQ1;
    *idum = IA1 *(*idum-k*IQ1) - IR1*k;
    if (*idum < 0)
    {
      *idum += IM1;
    }
    if (j < NTAB)
    {
      iv[j] = *idum;
    }
  }
  iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
  *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
  idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
  iy += IMM1;
}
if ((temp=AM*iy) > RNMX)
{
    return RNMX;
  }else
  {
    return temp;
  }
}
```

Annotated Bibliography

[1]

[2] “GPIO14 - General Purpose I2C I/O Expansion Chip.” [Online]. Available: http://www.robot-electronics.co.uk/htm/gpio14tech.htm

> This page gives an ovierview of the GPIO14 chip (which is essentialy a pre-programmed PIC16F818 8-bit microcontroller). Information on this page gave understand of how chip works and how it should be connected on I2C bus with other device. I used it heavily while migrating code from the Gumstix to RPI. In particular I was reffering to this page to understan what information is held in the different registers and where I need to write data to change chip behavior.

[3] “Raspbian Installer.” [Online]. Available: http://www.raspbian.org/RaspbianInstaller

> This page has extensive documntation describing how Raspbian OS should be installed on the RPi. It also contains suggestions for the tweaks

[4] “RIEtech Global.” [Online]. Available: http://www.rietechmotioncontrol.com/aboutus.html

> Includes description of the company: when it was established and what it produces.

[5] Dr Frédéric Labrosse, “Project: Idris.” [Online]. Available: http://www.aber.ac.uk/en/cs/research/ir/robots/idris/

> This web page gives an overview of the Idris vehicle and what it is used for. In particular I was interested in how PTU is mounted on the vehicle (its physical orientation).

[6] Gordons Projects, “Wiring Pi - GPIO Interface library for the Raspberry Pi.” [Online]. Available: http://wiringpi.com/

> This website is a home for the WiringPi library. I was consulting it when porting code from the Gumstix to the RPi. In particular I was looking how to install WiringPi library and what functions are available to send commands over I2C to the gpio14 chip.

[7] Jacob Fraden, *Handbook of Modern Sensors*. Springer, 2010, p. 680. [Online]. Available: http://www.plentyofebooks.net/2011/05/download-handbook-of-modern-sensors-4th.html 1441964657.

This book gave me understanding of why different sensors a not perfect and may have problems with accuracy. On 339 p. it covers gyroscope architecture and in particular discusses factors that affect their accuracy.

[8] Y. Kreinin, "How to mix C and C++." [Online]. Available: http://yosefk.com/c++fqa/mixing.html

[9] Loads, "Principles behind the Agile Manifesto." [Online]. Available: http://agilemanifesto.org/principles.html

Explains ideas behing the agile manifesto and what they really mean. Gave me an understanding of how development should be done in agile environment.

[10] NXP Semiconductor, "I2C-bus specification and user manual," 2012. [Online]. Available: http://www.nxp.com/documents/other/UM10204_v5.pdf

This user manual gives in depth description of I2C bus; I was interested in analysing the fundamental architecture of this bus as well as reading about SDA and SCL signals

[11] Raspberry Pi Foundation, "What is a Raspberry Pi?" [Online]. Available: http://www.raspberrypi.org

This website gives an overview of the Raspberry Pi computer and what it is capable of. It was a starting point for me in understanding of its hardware specifiacations and documentation about initial setup.

[12] Sagebrush Technology Inc, "Command Set Documentation, Pan-Tilt Gimbals,Servomotor Version," 2004.

This command set documentation defines a standard for the control of the PTU. I am interested in the contents of the tables with information about the error codes and general device control commands.

[13] Sagebrush Technology Inc., "Model 20 Pan-Tilt Gimbal User Manual," 2005.

This manual describes system set-up information and identifies the parameters necessary to operate a Model 20 Servo unit. In particular it describes available TASS stabilization commands.

[14] SensorWiki.org, "Gyroscope." [Online]. Available: http://sensorwiki.org/doku.php/sensors/gyroscope