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)

April 28, 2014
Version: 1.0 (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

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, so that if rover drives up the hill camera stays vertical. Theoretically to achieve camera verticality stabilization command could be invoked only once, when the rover is standing on the flat surface, or its position could be adjusted manually if it is on a curved surface and then stabilization command issued.

1.1 Problem Analysis

To ensure stability of the camera PTU uses gyroscopes, which in a perfect world would be enough. Unfortunately the well known problem of the gyroscopes is that their readings are affected by the different air temperature, magnetic effects, friction etc. [7]. In reality when PTU is issued with the stabilization command platform slowly drifts on both axes and in 1-2 minutes goes to the position which is 20 degrees different from where it should be. This project will concentrate on fixing this particular issue and try to enhance current system functionality.

1.2 Proposed Solution

One of the possible solutions could be the use of the additional PTU functionality. It allows to cancel drift by specifying, 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) and manually calculating new drift rate every 10 minutes becomes infeasible and tedious.

Proposed solution is to automate drift rate calculation so that it can be adjusted while driving. However this approach requires additional information about the current PTU orientation in the space (its inclination angles with respect to gravity). For this purpose can be used electronic inclinometer. Inclinometer response is electric signal representing an angle between the internal axis and the gravity vector [7]. Accurate reading with such device can be obtained only if it is in a stable position and does not accelerate, otherwise it will provide erroneous data representing sum of two vectors earth gravity and acceleration. This imposes certain limitations as to of when PTU

drift rate calibration can be performed, meaning that the vehicle will have to stop every time for this task to take place.

1.3 Control System

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

There was a decision made by the client to replace currently used platform (Gumstix). The rationale behind that were client concerns about the currently used Gumstix computer which is becoming old and in case of a breakdown it would be difficult to find a replacement parts. Another requirement 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. After a short discussion Raspberry Pi minicomputer was chosen as a replacement platform, it has enough power to compile code and all required interfaces to connect with other equipment.

PTU TASS library will be reused and its functionality extended to implement new features.

Server side and client side code will adjusted to provide access to the new functionality.

Main tasks

- Port current system from the Gumstix on to the Raspberry Pi.

  This includes recompiling code on the new system and in particular tweaking it to work with the I^2C

- Extend functionality of the PTU TASS library to incorporate new commands for the stabilization and drift rate.

- Implement drift rate calculation functionality.

1.4 Aims

Currently there is a system in place that provides functionality to control PTU and take readings from the inclinometer. The aim of this project is to modify current system which and extend its functionality to incorporate new features and abilities. The outcome of the project should be a system that provides simple way to control PTU including stabilization functionality.

Chapter 2

Design

2.1 Hardware Design

Proposed system will consist from the Raspberry Pi minicomputer(RPi), analogue inclinometer, gpio14 chip, a bunch of relays and the PTU.

2.1.1 RPi

As a main control unit by the client, where all the software will be running, was chosen the Raspberry Pi minicomputer. It will replace 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 and have all the required interface to be connect to the other peripheral. In particular it has GPIO pins including, SPI and I^2C interfaces, UART serial console, 5v and GND supply pins. Such powerful devices for this task may be an overkill, but decision was made by the client and I work with what I have.

2.1.2 Inclinometer

Inclinometer will be used to get data about the current chassis position in the space. Client suggested usage of the SCA121T dual axis inclinometer (figure 2.1) bolted to the chassis of the rover. It will provide information about the inclination angles during calibration. Since it is a analogue device we will need a A/D converter.

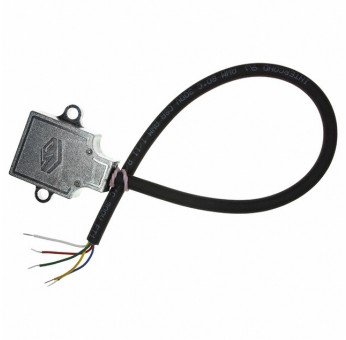

Figure 2.1: SCA121T-D05 Inclinometer

2.1.3 GPIO14 chip

The GPIO14 chip is a pre-programmed PIC16F818 micro controller. It is intended 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 switch on/off relays. This chip will be connected over the I^2C bus to the RPi.

2.1.4 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 typically is used to hold cameras, antennas or instrument positioning. In this projects it is used to hold a panoramic camera.

2.2 Software Design

My work will be based on the code that is already used on the Gumstix. Currently it is used to send control commands to the PTU, get readings from the inclinometer and switch on/off relays. The code base have three main parts in it. The client side software, server side software (running on the Gumstix and responding to client calls) and the library that implements TASS protocol.

2.2.1 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) was used RS-232-TTL converter. Currently this library has support for the basic commands like pan left/right, tilt up/down, get/set position, get/set rotation limits.

My task is to add implementation for the stabilize and drift rate commands as well as logic for the drift rate calculation.

2.2.2 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 current system from the Gumstix to the RPi platform and make it to work.

2.2.3 Clint Side

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

2.2.4 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. In particular it requires some configuration to prevent Linux from using the serial port and it also needs WiringPi library [6] to be installed to successfully communicate over the I^2C bus with the GPIO14 chip. All necessary configuration is covered in Appendix A.

2.3 Overall Architecture

2.4 Some detailed design

2.4.1 Even more detail

2.5 User Interface

2.6 Other relevant sections

Chapter 3

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on real users? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

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] 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

[10] 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.

[11] 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.

[12] 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.

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