Progress Report

# Introduction

The Global Positioning System (GPS) is a network of satellites that allows calculation of positional data using triangulation. Distances are calculated by measuring the travel-times of light-speed messages. Naturally, a good measurement of position relies upon the performance of the clocks.

With multiple on-board atomic clocks [1], satellites keep time with great accuracy and precision[[1]](#footnote-1). However, the timing output of the GPS receivers is truncated to the nearest second. The aim of the project is to generate sub-second timing from this output. Such a device would be useful for precise, consistent timing in multiple locations.

This involves relating properties and patterns of the receiver’s output signal to the actual time.

# Methods

While there are a variety of mechanisms for presenting the GPS information, we concern ourselves with those that send out data in *serial* format. In this system, data is encoded by voltage in binary format which is sent one *bit* at a time. These signals can be easily detected and decoded by a microcontroller.

Here an *Arduino Uno* microcontroller was used alongside the *Adafruit Ultimate GPS* receiver, specifically designed for the Arduino. This microcontroller is capable of reading and writing serial, measuring voltages, and writing to SD card. It also houses an internal oscillator clock for timing.

# Preliminary Tests

The Arduino’s internal oscillator is key to interpreting the receiver’s output patterns since it is used for all timing measurements. The first task was to assess its performance; how accurate and precise is the clock?

The Arduino was programmed to relay its time measurement periodically to a computer, which compared the difference between measurements to its own clock. Without testing the computer’s time-keeping it is not possible to assign concrete values to the Arduino’s clock; instead the test was designed to give rough quantitative bounds to the microcontroller’s timing. Additionally, in order to minimise the effects of latency in sending and receiving signals (of order milliseconds) the time period for sending time updates was set to minutes.

According to the computer’s clock the Arduino measured one-minute intervals with a relative accuracy of 0.999807±0.000013 and five minute intervals with an accuracy of 0.999812±0.000003, suggesting that the Arduino’s clock should be accurate to within a few parts per ten-thousand, with an uncertainty that perhaps scales like for measurements of time . Then the uncertainty for one second should be within a millisecond.

With a rough idea of the timing performance of the Arduino, we moved on to the GPS receiver with the intention of deducing how regularly the GPS receiver sends out its messages. At first the arrival time of each message was measured and relayed to a computer via USB, but was later changed to a more convenient method of saving to SD. The GPS receiver sends out a message (known as a *sentence*) every second containing a variety of information including time and position. For now just the time of arrival was of interest.

Plotting the time difference between messages

|  |
| --- |
| D:\Users\Duncan\Documents\Uni\Uni Work\Work\PHYS 6006 Project\Atomic-Time\GPSDrift\Initial\Plots\231312 18102015 GPSDrift_measurementsserser_dT_dist.png |
| Figure 1  Distribution of time between GPS serial data |

Figure 1 shows the distribution of time differences between the GPS receiver’s outputs. While the series of peaks forms an interesting pattern[[2]](#footnote-2) it is not the primary concern; of most interest is the large spread in the time between GPS output. One might think that the GPS doesn’t wouldn’t keep time well with so much variance in times. However, the large and negative correlation between consecutive time difference in Figure 2 shows that longer time differences are generally immediately compensated by subsequent shorter time differences and vice-versa; there is an underlying mechanism that keeps better time that the GPS serial output.

|  |
| --- |
| D:\Users\Duncan\Documents\Uni\Uni Work\Work\PHYS 6006 Project\Atomic-Time\GPSDrift\Initial\Plots\231312 18102015 GPSDrift_measurementsserserCorFn.png |
| Figure 2  Self-correlation function of GPS serial data times. Reflects how well-correlated the times of a given serial message and a serial message order samples in the future are |

The sporadic nature of the time differences foreshadows a problem that must be overcome; with a spread of approximately 50 ms either side of the mean value, the time difference between GPS outputs and the turn of the second must have a spread of at least 25 ms either side of the average. In other words, measuring the time of arrival of one GPS message would give an uncertainty of *at least* 25 ms to the actual time.

A perhaps more useful question to ask is *how long after the second turns are the messages sent?* In order to answer this question one needs an accurate clock for reference. The Adafruit GPS receiver has a built-in pulse-per-second (PPS) output; on the turn of every second it outputs a high voltage before returning to low. This can be detected by the Arduino, and so the times between the start of the pulse and the serial messages can be measured.

Measurements of the PPS indicate that its period is equal to one second to within one part in ten-thousand.

Measuring the time difference between the PPS and serial output one finds

|  |
| --- |
| D:\Users\Duncan\Documents\Uni\Uni Work\Work\PHYS 6006 Project\Atomic-Time\Interrupt + NMEA\Results\GPSMIL12ChckdCorppsser_dT.png |
| Figure 3  Time-domain PPS-serial differences |

Figure 3 demonstrates the semi-chaotic nature of PPS-serial values. For regions of approximately one-thousand seconds the PPS-serial difference forms a regular structure of two bands separated by 50 ms. Between these sections the value fluctuates wildly.

|  |
| --- |
| D:\Users\Duncan\Documents\Uni\Uni Work\Work\PHYS 6006 Project\Atomic-Time\Interrupt + NMEA\Results\GPSMIL12ChckdCorppsser_dT_dist.png |
| Figure 4  Distribution of the time between the start of the PPS and the start of the serial data |

The distribution Figure 4 shows the overall trend between PPS and serial arrival times. The width of the main peak is 200 ms

# Bibliography

|  |  |
| --- | --- |
| [1] | “GPS Timing,” GPS.gov, [Online]. Available: http://www.gps.gov/applications/timing/. [Accessed 16 Dec 2015]. |
| [2] | “Nist Time FAQ,” NIST, 04 Dec 2015. [Online]. Available: http://www.nist.gov/pml/div688/utcnist.cfm. [Accessed 16 Dec 2015]. |

1. However the timings are inconsistent with the widely used UTC clocks. This does not affect the timing performance of the GPS [2] [↑](#footnote-ref-1)
2. The pattern indicates a quantisation of around 10 ms in either the GPS’s output or the Arduino’s clock. Further analysis of the Arduino’s timekeeping ability suggests that the source of these peaks is the GPS. [↑](#footnote-ref-2)