2010 Intel Cup Embedded System Design Contest

Improving Exercise Bike Experience with Google Street View and Virtual Reality

28 June, 2010

The Chinese University of Hong Kong

Faculty Mentor: Prof. Xu Qiang

Author: Chan Chun Wing, Chan Hiu Yu, Zhang Qi

Abstract:

Cycling is one of the best forms of exercise available. As cycling is often affected by the weather and available places, many people prefer cycling at home with exercise bikes. However, prolonged cycling on an exercise bike could be very boring. The aim of this project is to provide a better exercise bike experience. The user can cycle on any road around the world with the Google Street View scene updating according to the cycling speed detected by the encoder. With digital compass, the gyroscope and the accelerometer working together, the computer can detect the movement and orientation of the head and update the scene in real time. With video goggle, a virtual reality experience is thus achieved.

Keywords:

exercise bike, virtual reality, google street view, accelerometer, gyroscope, compass

Declaration of Originality

We hereby declare that this thesis and the work reported herein was composed and originated entirely by ourselves. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given in the references.

Team Members Signature: _____

Name (in Block Letters): CHAN CHUN WING, CHAN HIU YU, ZHANG QI

Date: 28 June, 2010

Table of Contents

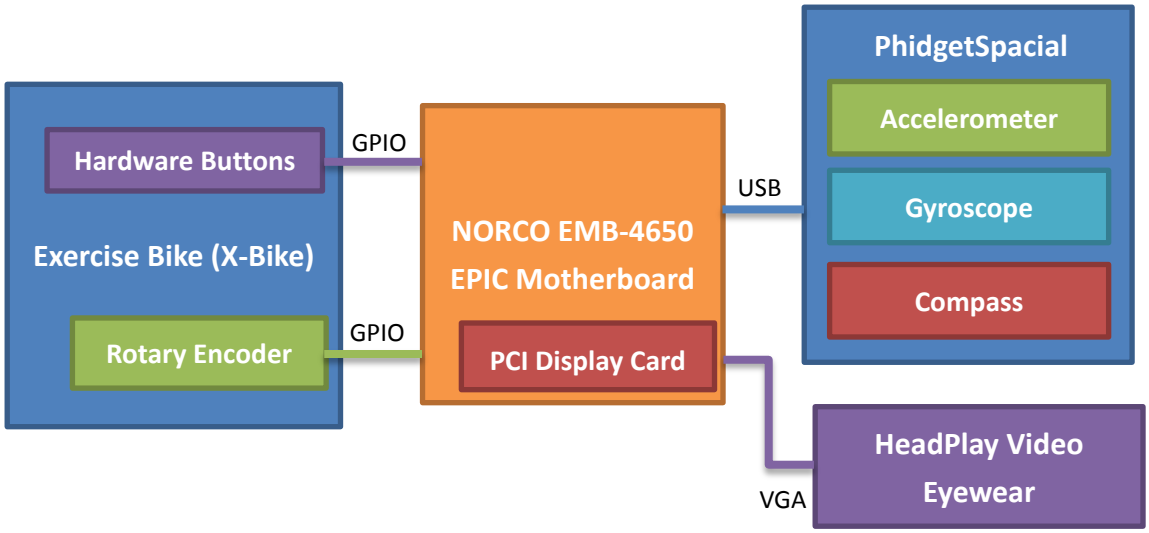
1 Introduction

1.1 Background

Cycling is one of the best forms of exercise available [1]. It is proved that cycling is effective in reducing extra calories, speeding up metabolism and training body coordination.

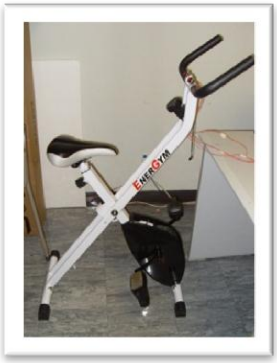
Due to the fact that cycling is often affected by the weather and requires a lot of space, many people prefer cycling at home with exercise bikes. However, prolonged cycling on an exercise bike could be very boring. The aim of this project is to solve this problem by improving the exercise bike experience and therefore promote health by attracting more people to take part in cycling.

1.2 System Schema



1.3 System Overview

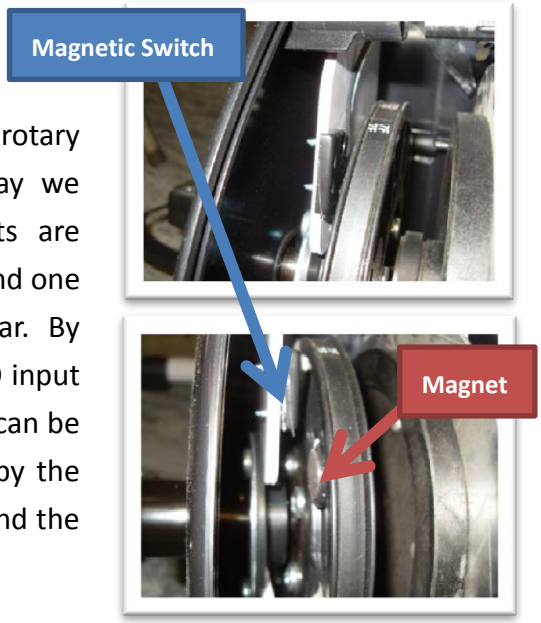
1.3.1 Exercise Bike (X-Bike)



We have chosen the X-Bike which is a low-priced and small-sized exercise bike for our project. Unlike the more expensive exercise bikes in the market, the X-Bike is not equipped with extra features like monitors, heartbeat sensors and training programs. However, we found the

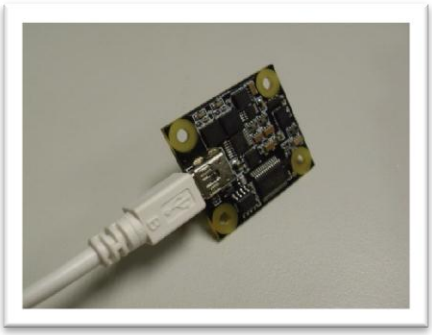
simplicity an advantage as we can install our own system and attach other devices to it. For example, some hardware buttons are installed and connected via GPIO so the user to change direction quickly when cycling.

1.3.2 Rotary Encoder



To detect the cycling speed, we installed a rotary encoder in the gear of the bike. The way we design the encoder is that four magnets are placed evenly distributed around the gear and one magnetic switch is placed above the gear. By connecting the magnetic switch to the GPIO input of the motherboard, four on-off transitions can be detected when one rotation is performed by the gear. Through some calculations we could find the cycling speed in terms of metres per second.

1.3.3 Accelerometer, Compass and Gyroscope



To detect the movement and orientation of the head we used PhidgetSpatial which is a single chip equipped with an accelerometer, a compass and a gyroscope. With these three components we managed to measure at the same time the acceleration, magnetic field and angular rotation in the 3D space. The chip is connected via USB to the motherboard.

1.3.4 Video Eyewear

Besides the PhidgetSpatial, to achieve the virtual reality effect the user will wear a pair of HeadPlay Video Eyewear. Through the video eyewear the user can enjoy a huge-screen experience with minimal space. We hope that with virtual reality implementation the user will have a realistic feeling that he or she is cycling at the place shown in the Google Street View.

Below is a summary of all the peripheral devices used in our project:

Name	Function	Port
Exercise Bike	Control Streetview	N.A.
Hardware Buttons	Change Direction	GPIO
Magnetic Rotary Encoder	Get cycling speed	GPIO
PhidgetSpatial 3/3/3 (Accelerometer, Gyroscope and Compass)	Detect tilt and orientation of user's head	USB
HeadPlay Video Eyewear	Show Street View Scene	VGA

1.3.5 Software

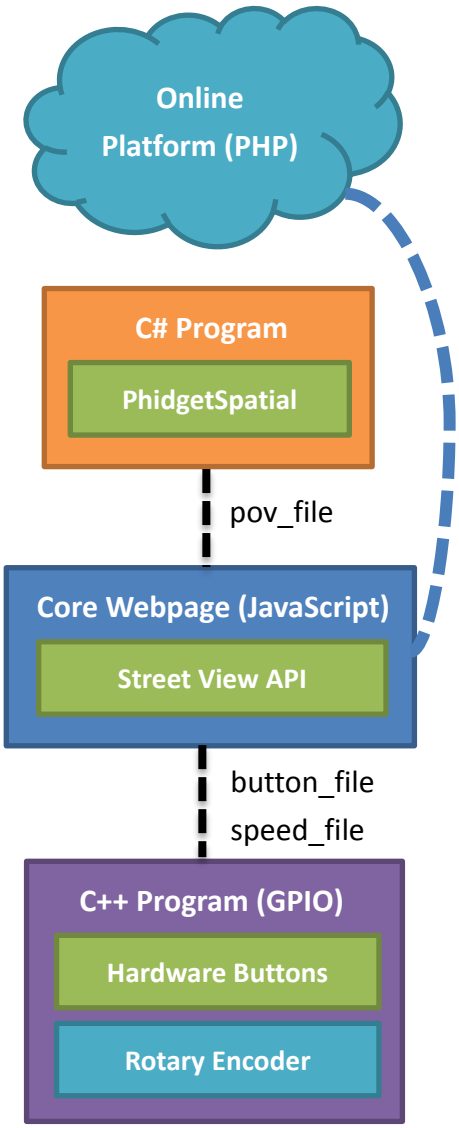
The diagram on the right shows the interaction of different programs we designed for this project. The C# program is responsible to get the compass, gyroscope and accelerometer value returned by the PhidgetSpatial.

The C++ program is responsible to get the GPIO status. Both the hardware buttons and rotary encoder are connected to the GPIO pins of the motherboard.

The webpage written in HTML and JavaScript is the core of the project. It handles user input and show the Street View pictures.

These programs communicate with each other through writing and reading text files.

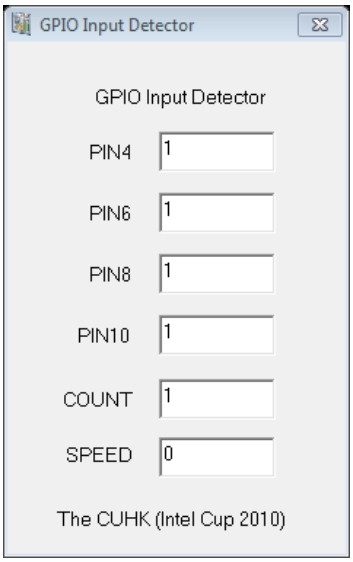
The online platform written in PHP is implemented and hosted as a separate webpage. The online platform supports various interactive features. For example, the core webpage can submit user scores to the platform through the Internet.



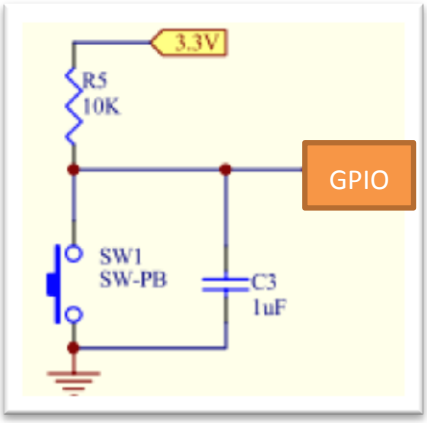
2 Design and Implementation

2.1 Major Components

2.1.1 Rotary Encoder

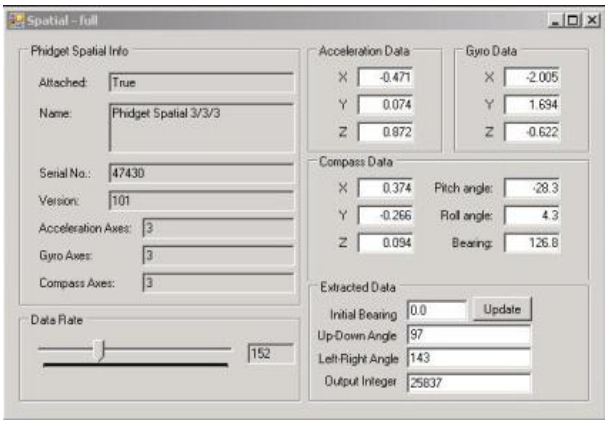


The rotary encoder is set up to record the actual distance the user has cycled. As the user is cycling, data will be sent to the GPIO program. Our primitive design is that only one magnet is installed at the gear, which is the part linked to the pedal. Nevertheless, the experimental result obtained was not satisfactory as only one transition is given per rotation. To fix this problem, we installed four magnets around the gear in order to detect smaller changes in rotation.



The diagram explains how the rotary encoder works. A magnetic switch is installed above the gear. The switch is disconnected when a magnet passes through it. Approximately 3V will be recorded by the GPIO (General Purpose Input Output) pin on the motherboard. The GPIO program at the background will record the number of transitions every 1 second to calculate the current rotation speed.

2.1.2 PhidgetSpatial (Accelerometer, Gyroscope and Compass)

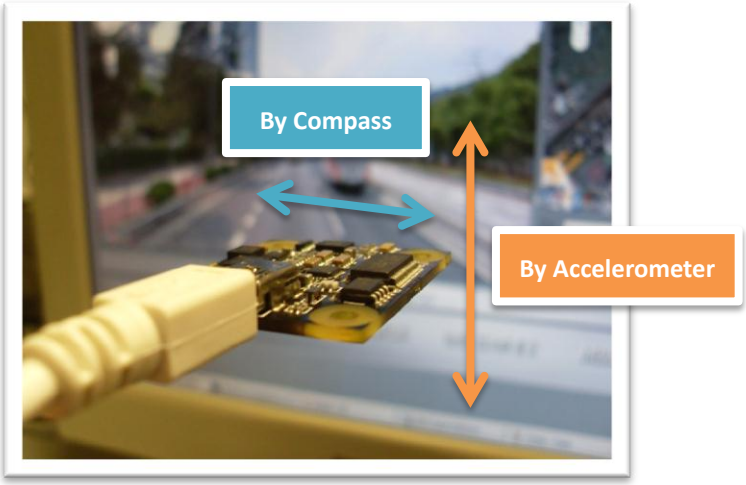


To make use of the 360° viewing feature provided by the Google Street View API to achieve virtual reality, we need to detect the movement and orientation of the user's head. To do this we used the PhidgetSpatial which is composed of an accelerometer, a gyroscope and a

compass. It is connected through USB to the motherboard. The accelerometer is responsible to detect vertical motion while the gyroscope and the digital compass are used to detect the horizontal motion.

Accelerometer

As we have mentioned before, the accelerometer is responsible for detecting vertical motion. This is a device measuring the acceleration along the x, y, and z axes with respect to the free-fall gravity. The accelerometer acts like a spring. When the user is looking upward, it will be reflected in the change in of y-axis value and it is captured by our C# program. The program continuously reads the data from PhidgetSpatial and writes into a text file every 0.1s. Since the accelerometer is only capable of determing the motion respective to the free-fall motion, it is unable to spot the horizontal motion, hence a gyroscope and a digital compass are introduced.



Digital Compass (Magnetometer)

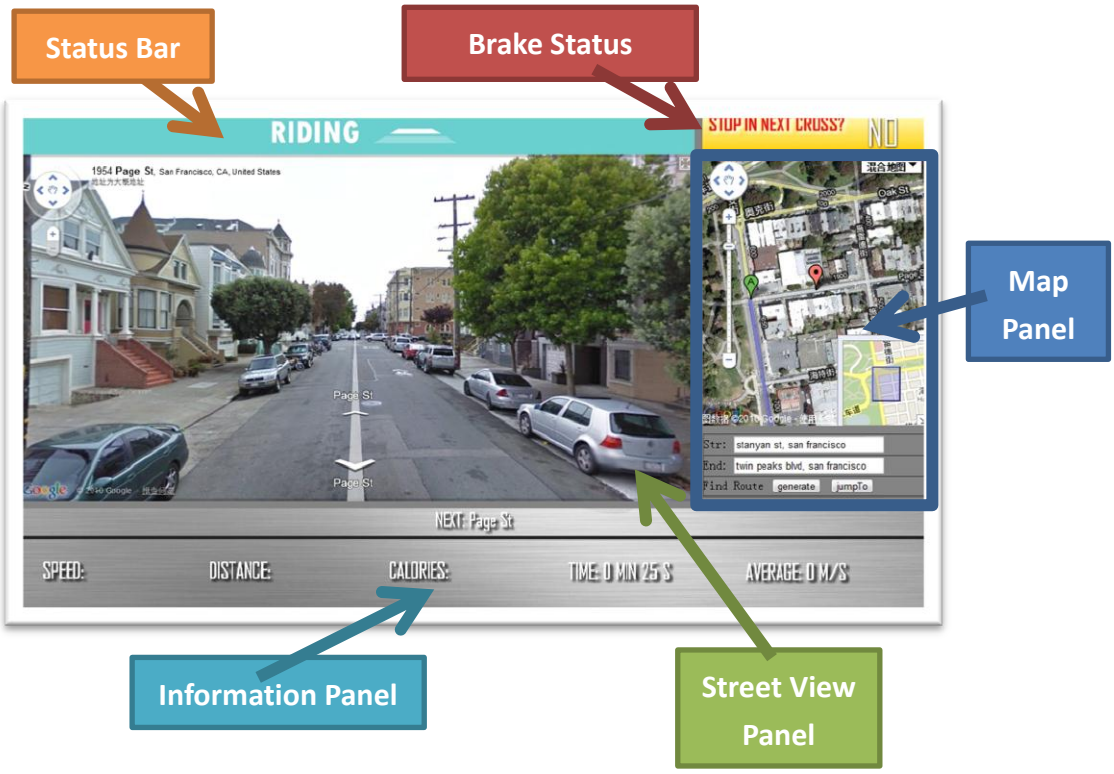
The digital compass is able to detect the change in magnetic field. In other words, it is able to spot out the changes in horizontal angular rotation. This device is used together with the accelerometer to facilitate the up-down-left-right motion detection to achieve virtual reality effect. Once a motion is detected, our C# program will calculate the compass bearing which the user is looking at mathematically and update the file with the calculated result.

Gyroscope

The digital compass alone has several problems when used on moving platforms as it must be level, and it tends to correct itself rather slowly when the platform turns. To solve the problem, a gyroscope is also introduced. The gyroscope is a device for measuring orientation, based on conservation of angular momentum. With the gyroscope, even the compass is tilted, we can also adjust to the correct compass bearing through some matrix calculation performed in the C# program.

2.2 Core Webpage

2.2.1 Introduction to Webpage Interface



Status Bar

The status bar shows the current state of the Google Street View panel. For example, it tells the user to wait if it is building the graph or warn the user if a crossing is coming.

Street View Panel

This panel is for displaying the Street View for user. This panel will support 3 functions: real time sight of the user, turning and cycling ahead.

Map Panel

The map panel acts as a guide and control for the user. The user can see the current location, generate a route or move to another place.

Information Panel

The name of the next road is shown at the top so that the user can know which road he/she is turning into. Other information like the current speed, distance travelled and calories lost are also shown.

2.2.2 Google Street View Implementation

Introduction to Google Map API

The Google Map API [2] provides discrete but non-continuous view points for users and developers. All the points are located step by step and each of them is not just a simple photo but a 360° panorama.

Graph Construction

Google does not provide offline Street View pictures but only provides basic information like location, panoID (a special string to represent a view point), links (relationship between adjacent view points). In order to speed up, our system downloads the information of the viewpoints in a certain area when the webpage is loaded. The information are stored into a graph which every node represents a viewpoint, and every edge represents “linked” or “reachable” relationship between the nodes.

For the implementation, we adopted the DFS (Depth First Search) algorithm. Through setting a bound, we are able to construct a graph with proper size. A new graph will be built when the user reaches the boundary of the graph.

Buffer Implementation

As mentioned above, offline photos are not provided by Google. The speed of a typical bicycle can be as fast as 20m/s. As each Street View picture is separated by 10m, our system needs to be fast enough to switch a picture every 0.5s in order to reach a moving speed comparable to a real bicycle.

Through investigation we found that the bottleneck of the system lies in the overhead in sending frequent requests for Street View pictures to and from the Google server. Understanding this reason, our solution is to open four panorama windows so that we can send four requests at a time.

In the beginning, the current location is displayed. Meanwhile the next 3 points are also loaded by other three buffers in backstage. We found the performance of the webpage improved drastically after this implementation is deployed.

Riding Function

The buffer approach is particular useful when the bicycle advancing at high speed. When the user advances by one step (10m), our system will show the buffer of the next location. Meanwhile, the current buffer will jump to the location 4 steps away. The following table illustrates how the buffer works.

STEP	STATE OF BUFFER	STEP	STATE OF BUFFER
Step0	**[A-0]** B-1 C-2 D-3	Step4	**[A-4]** B-5 C-6 D-7
Step1	A-4 **[B-1]** C-2 D-3	Step5	A-4 **[B-5]** C-6 D-7
Step2	A-4 B-5 **[C-2]** D-3	Step6	A-4 B-5 **[C-6]** D-7
Step3	A-4 B-5 C-6 **[D-3]**	Step7	A-4 B-5 C-6 **[D-7]**

Turning Function

Two hardware buttons are installed on the bicycle. The left one is for turning left, and the right one is for turning right. The status of the buttons are fetched by the GPIO program every 0.5s and written into a text file which will be read of the webpage through JavaScript.

The user can select the road they want to advance using the hardware buttons. Before each crossing, the webpage will warn the user to make sure that the user will not miss it.

POV (point of view) Real Time Tracking

This function is controlled by the PhidgetSpatial chip. The system reads two numbers which are yaw and pitch respectively from the text file periodically. Yaw represents the horizontal orientation while pitch represents vertical tilt.

The Street View scene is updated whenever these two numbers change. The video eyewear ensures that the user can see the Street View scene no matter how the head turns. Thus, a virtual reality effect is achieved as the scene reflects the changes of user's POV realistically.

2.2.3 Google Map control and display system

User Tracking Marker in the Map

There is a red marker showing the current location of the user in the map panel at the right of the interface. When advancing, the marker will track the movement of user. It is implemented using the functions provided in the Google Map API.

Drag-able Jumping Function in the Map

A convenient feature is provided for user to choose the location that they want to start their ride. The user can drag the red marker to anywhere in the world that has Google Street View available.

When user drags the red marker to a place, the system will find the location parameter (e.g. Stanyan St, San Francisco) at the point. The parameter is then passed to the Google API to fetch the corresponding Street View picture.

Automatic Route Generation

Below the map there is a panel with two text boxes. The user can input the starting and ending locations and generate a route automatically. The route will then be displayed on the map. The Jump-To button will reset the Street View panel and show the picture of the desired starting location.

To implement the function, we make use of the class “GDirection” provided by the Google Map API. It will generate route just by the starting and ending addresses which are two strings.

3 Test Plans and Result Analysis

3.1 Test 1 – Rotary Encoder

Originally, we adopted the original design of the rotary encoder which employs only one magnet inside. After various testing, we have found that the result obtained using only one magnet was not good enough. Therefore, we decided to conduct an experiment to find the optimal number of magnets that should be used for the encoder.

Procedure

1. Install one more magnet for the rotary encoder
2. Start cycling at a constant speed for one minute
3. Measure the number of transitions recorded by the GPIO program
5. Evaluate the result.
6. Repeat Step 1 if necessary.

Results

Number of Magnet used	**1**	**2**	**3**	**4**	**5**
Number of transitions recorded in 1 minute	62	124	186	248	Fail

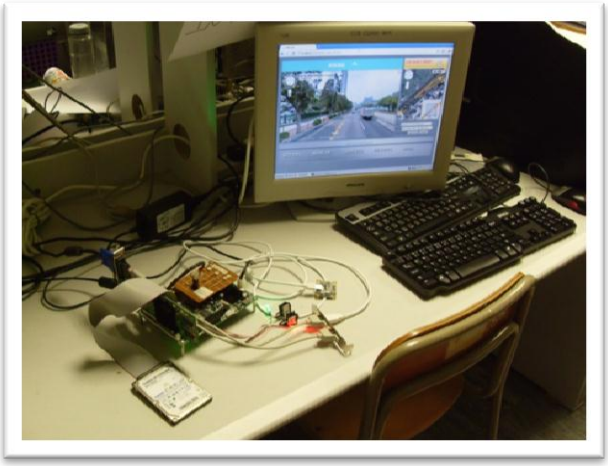
Analysis

We found that using more magnets for the rotary encoder is better. A more accurate average speed can be calculated as there are more transitions in the same period of time, thus it can detect small change in distance travelled. However, when the number of magnet is increased to five, the magnets are placed too near each other and the magnetic switch does not work anymore. Therefore, we found the optimal number of magnet that should be used is 4.

3.2 Test 2 – CPU Performance

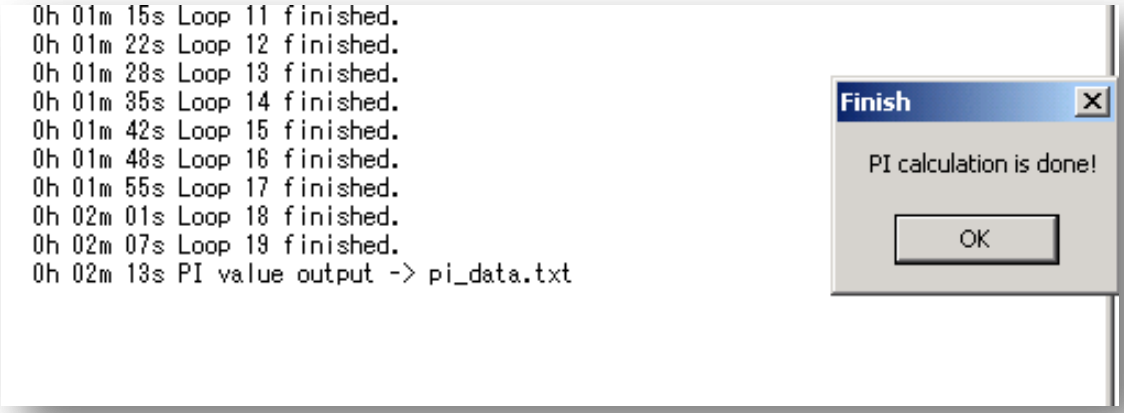
To find the capability of the Atom Z510P CPU on the board, we have used several readily available tools to explore the performance of our CPU.

Procedure



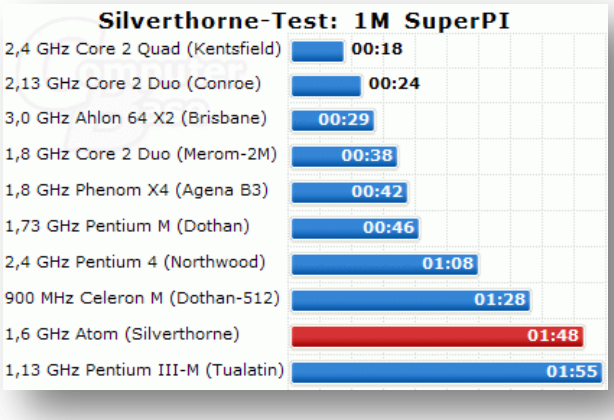
Use SuperPI as a tool to evaluate CPU performance. We aim to evaluate the performance of our CPU by measuring the time that the software needs to calculate PI value up to 1M digits.

Results



The Atom Z510P CPU takes 133 seconds to calculate the value of PI up to 1M digits. The result reflects that the performance of the CPU is not very high.

From the chart [3] shown on the right, we can see that the performance of Atom Z510P is not comparable to any desktop or laptop CPU in the market. At similar clock speed, the Pentium III-M is still 14% faster.



Analysis

Even our experiment shows that the performance of the Atom Z510P CPU is not as good as those we use every day, there are good reasons why this CPU is suitable for our project. The Atom Z510P has very low power consumption and is very cool even when full-loading. When we are designing an embedded system, the size, quietness of system and power consumption are all very important factors. In this sense, the Atom Z510P is very suitable. However, due to the processing lower speed, we should also adapt our design to avoid very complex calculation by choosing some lighter algorithm.

3.3 Test 3 –Smoothness when Interchanging Photos

As mentioned above, we used a buffering technique to improve the performance of loading the Street View pictures. We conducted a test to find out the optimal number of buffer needed.

Procedure

We change the number of buffer used for pre-downloading picture in the background and measure the number of seconds taken to change to another viewpoint.

Results

Number of cache photos	Seconds taken changing to another photo
0	1s
2	0.5s
4	0.2s
6	0.3s

Analysis

Form the experiment we found that the optimal number of buffers should be 4. When the number of buffers is below 4, increasing the buffers can improve the performance as the bottleneck lies in the overhead of the internet.

However, when the number of buffers has reached 4, further increasing the number of buffers will have no positive effect on the performance. The reason is that the bottleneck now lies on the CPU performance. There is not enough computing power for the CPU to handle so many Street View panoramas as the embedded flash object for Street View is highly resource-intensive.

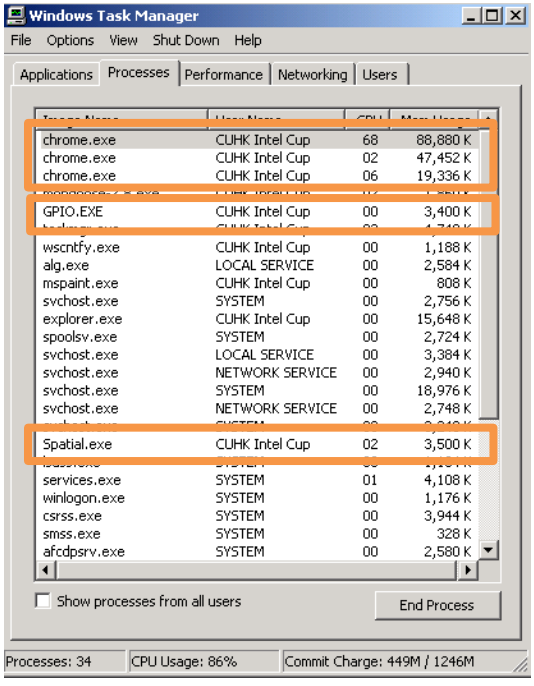
3.4 Test 4 –System Utilization

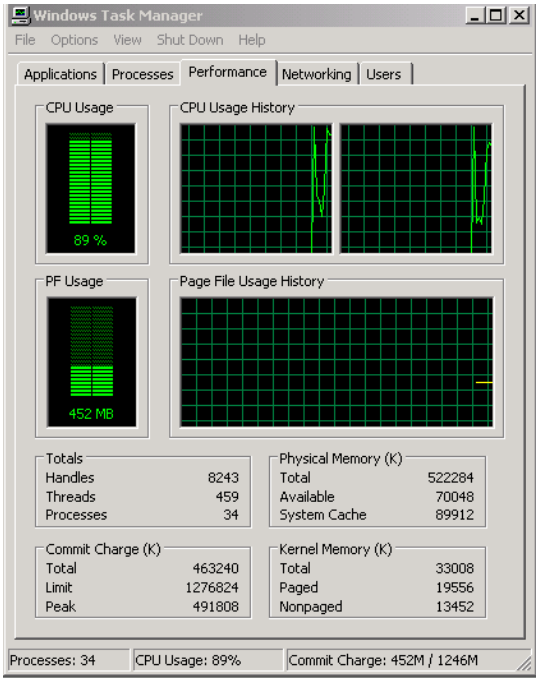
After we have done the programming, we need to test whether the system is powerful enough to support our programs. The following test has been performed to find the system utilization when our all programs are executing.

Procedure

1. Start the GPIO program, PhidgetSpatial program and the core webpage.
2. Check the CPU usage and occupied memory of each program using Windows XP's Task Manager.

Results





Analysis

From the picture on the left, we found that the Google Chrome browser consumes the largest amount of CPU resources and memory while the GPIO program and the PhidgetSpatial program only contribute a small amount of utilization. From the picture on the right, we found that we have used nearly 90% of the CPU resources. The reason for the high consumption is that the flash player of Google Chrome needs a lot of resources to continuously update the Street View panel. Overall, our design has made use of almost all computing power of the Atom Z510P.

4 Discussion

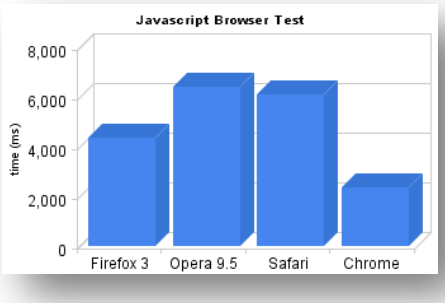
4.1 Improving I/O performance

The only communication channel between the GPIO program, the PhidgetSpatial program and the Street View webpage is by writing and reading text file. As we need to write GPIO status and the value returned by the PhidgetSpatial at 100ms interval, the I/O overhead could be very large if we store the file in a hard drive. To reduce the overhead to minimum, we set up a 64M ramdisk using the software Gavotte Ramdisk in windows. We put the browser cache and text files for communication between programs into the disk as they are most frequently accessed. As the access time and reading speed of ram is much higher than hard drive, the performance of the system is improved.

4.2 Improving overall system performance

Further optimization is used in the operating system in order to achieve highest performance. For example, turning off unnecessary visual effects, disabling some services like event logging and applying tweaks to the registry can help save system resources.

Besides, choosing the most appropriate browser is also a very critical matter. According to the benchmark [4] shown on the right, Google Chrome executes JavaScript 50% faster than Firefox. Therefore, we have chosen Google Chrome as our browser.



5 Possible Development

5.1 A more complete online platform

The online platform we have implemented is primitive. Functions like allowing users to share their time and distance travelled on Facebook can be added. Also, users can invite their friends to ride in a race using the platform. The platform should make use of the internet and online social network to attract users to use our machine.

5.2 Wireless connection between components

The HeadPlay Video Eyewear and Phidget Spatial are connected through the VGA and USB respectively to the computer. When user is cycling, it is possible that these cables may put burden to the user. It is better if we can replace it with a wireless link. However, there is no commercial product available yet that can transfer VGA signals wirelessly.

5.3 Smoother transition between pictures

Google Street View pictures are taken once every 10 metres. Therefore, a "discontinuous" feeling may arise when the user is cycling and pictures are switching. To alleviate this problem, one possible approach is to use a complex image morphing technique and create a smooth transition between images.

6 Conclusion

Riding an exercise bike could be boring. Our system aims to enhance the exercise bike experience by introducing Google Street View and virtual reality. User can feel like they are cycling on the real road as the scene continuous changes in real time according to the cycling speed and head movement of the user. We hope that our system will be able to encourage people to do more exercise.

7 References

[1] Derek Both. streetdirectory.com. [Online]. http://www.streetdirectory.com/travel_guide/40132/recreation_and_sports/the_advantages_of_cycling.html

[2] Google. (2010, June) Google Map API Developer Guide. [Online]. http://code.google.com/apis/maps/documentation/javascript/basics.html

[3] (2008, March) Computer Base. [Online]. http://www.computerbase.de/news/hardware/prozessoren/intel/2008/maerz/erster_benchmark_intels_silverthorne/

[4] Mohamed Mansour. (2008, September) m0 | interactive. [Online]. http://www.m0interactive.com/archives/2008/09/03/google_chrome_javascript_benchmark_test.html