



Sun Shuo

USER INTERFACE OF RAAHE MUSUEM

USER INTERFACE OF RAAHE MUSEUM

SUN SHUO
Bachelor's thesis
Spring, 2013
Information Technology
Oulu University of Applied Sciences

.PREFACE

This project was designed for Raahe Museum. The target was to show some areas and history of Raahe, Finland. The work supplies a graphical user interface which is projected on the curtain to visitors. They will enjoy the wonderful world of Raahe. Visitors can consider themselves as an explorer.

Raahe 1 May 2013/5/21

Sun Shuo



ACKNOWLEDGMENTS

Here I would like to thank VTT technical research center of Finland for providing the devices I need. And Raahel Museum gave me an opportunity to design and set it as my Bachelor's thesis topic. This was a real experience for me to use all things I have learnt during my 4-year studies.

Next, I want to thank my supervisor, Risto Korva. He helped me to analyze the structure of this project when I was confused. He drew a basic diagram of the hardware connections and layers. This diagram told me the basic running processing and principals, especially the data transmitting direction among PC, sensor board controller and sensor board.

Also, I would like to thank my electronic and hardware teacher, Juha Rätty. He supplied me the knowledge of the components in the IC board and usage.

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Degree Programme of Information Technology

Tekijä Sun Shuo

Opinnäytteen nimi: User Interface of Raahen Museo

Työn ohjaaja: Risto Korva

Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä 49

Työn teettäjänä oli Raahen museo. Työn tarkoituksena oli suunnitella ja toteuttaa käyttöliittymäjärjestelmä, jolla mallinnetaan arkeologisia kaivauksia Raahen ympäristössä.

Tärkein osa työstä oli suunnitella ohjelma, joka huolehtii kommunikoinnista tietokoneen ja kapasitiivisen sensorilevyn välillä. Sensorilevy koostuu 60*60 cm:n kokoisesta sensorikalvosta ja piirilevystä, jolla on sensoripari ja mikrokontrolleri Atmel's ATmega 32L. Piirilevyllä on kapasitanssin mittausspiiri (AD7147) ja sarjaportti - USB - muunnin (FT232R). Sensorilevy tunnistaa kädensijan jopa 10 cm:n etäisyydellä. Tietokoneen kommunikointiohjelma lähettää sensorilevylle jatkuvasti useita kertoja sekunnissa komennon, johon sensorilevyn mikrokontrolleri vastaa lähettämällä kädensijan koordinaatit.

Käyttöliittymäohjelman tehtävänä on ohjata käyttäjälle tulevia ilmoituksia ja vaihtaa kuvia ja ääniä. Kuvien vaihto tapahtuu kun käyttäjä tekee kädellä kaivamista matkivia liikkeitä. Näin simuloidaan yhä syvemmälle etenevää arkeologista kaivausta.

Työssä saatiin toteutettua toimivat kommunikointi- ja käyttöliittymäohjelmat, jotka toteutettiin C++ -kielellä. Näitä voidaan hyödyntää museon muissa vastaavissa järjestelmissä.

Avainsanat: C++, MFC, USB, sarjaportti, kapasitiivinen sensorikalvo

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author : Sun Shuo

Title of Bachelor's thesis: User Interface of Raahe Musuem

Supervisor : Risto Korva

Term and year of completion: Spring 2013 Number of pages: 49

This Bachelor's thesis was commissioned by Raahe Museum. The aim was to develop a user interface for the museum and to control the UI by some external devices. It is like a window for visitors who come into the museum. They could observe some mysterious areas of Raahe city.

The most important thing was to set up communications between a laptop and a sensor board controller, called a position machine program. A laptop or a desktop needs to send a command to a circuit board in real-time. And after receiving it, a microcontroller calculates the coordinate of a user's hand above the sensor board by the changing capacitance value, then it transmits it to a laptop or a desktop, and the next thing is the UI development. It holds the same programming language as a position machine. This UI could implement picture switching, information windows showing and also playing some sounds. External devices support UI controlling. They are Capacitance-to-Digital converter(AD7147), Atmel's ATmega 32L, Serial to USB format converter(FT232R), and a flexible sensor board.

Finally, the work could display a diagram UI successfully with a sensor controller board and a capacitance sensor board. It is possible for museum persons if they need to change the picture and sounds in the UI program. It becomes a new UI program.

Keywords: C++/MFC, serial port communication, capacitive sensor user interface.

TABLE OF CONTENTS

PREFACE	3
1 INTRODUCTION	8
2 DEFINITION	10
2.1 External devices description	10
2.2 Requirements	12
2.3 Important components in communication	18
3 Implementation	21
3.1 Communication	21
3.2 User Interface	24
4 TESTING	29
5 POSSIBILITIES OF FURTHER DEVELOPMENT	30
5.1 Changing picture	30
5.2 Changing coordinate recognition	30
5.3 Changing threshold	30
5.4 Changing music	31
6 LIST OF REFERENCES	32

1 INTRODUCTION

Nowadays, a microcontroller is generally used in electronic field. Such as ATMEL, Philips, MicroChip. They can be used in different industries, even in toy manufacturing.

This project contains two parts: An user interface application and external devices (a spant capacitive sensor, an AD7147 capacitance-to-digital converter and an ATmega 32L circuit board). The most important was to set up a bridge between a PC and a microcontroller. So the first step was to complete a PC Circuit board application.

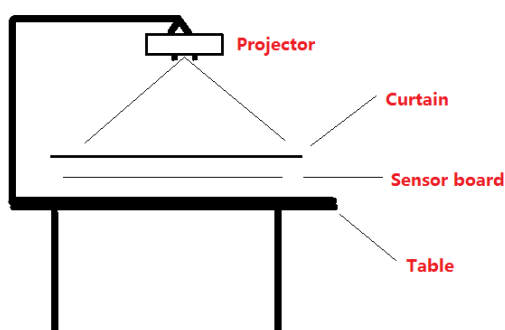


FIGURE 1. Sectional display view

When a visitor's hand moves above the sensor board, the capacitance value of that point is different from the average value (the average value is 32768 without objects above). Then the microcontroller calculates the coordinate of the hand position and sends it to the PC.

The UI contains two options. One is called graveyard. If users move hand above this option, the panel will be switched to a map in a cartoon type. Some grave points are shown in the map. Users can also move a hand to these points. Some of the points supply the information of a current grave. And for one point, the panel will be turned to the ground surface. Then users can do digging actions on the curtain. After several actions (UI program counts digging actions), the counting value arrives 15. A deeper layer can be seen, as the action is going on, deeper and deeper. Finally, if you are lucky

enough, a metal sword or a bronze ring comes into the view. That seems an award for users.

The first task was to make a program (HOST COMPUTER), which sends a command as an ASCII code to the sensor controller board and receives a coordinate calculated by it,. Then it makes some sense to the UI program, which is the second step.

The configuration is shown in the figure 2:

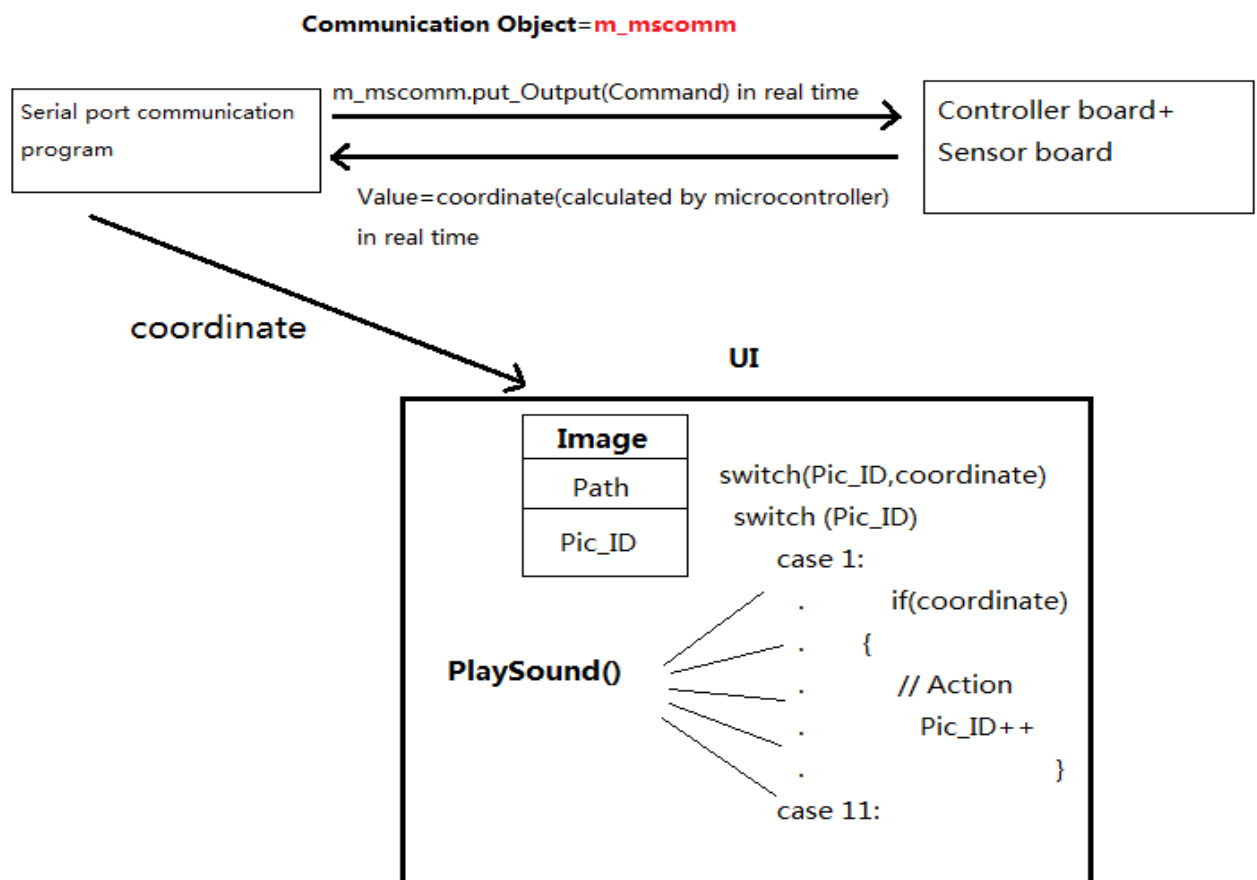


FIGURE 2. Configuration of UI part

2 DEFINITION

2.1 External devices description

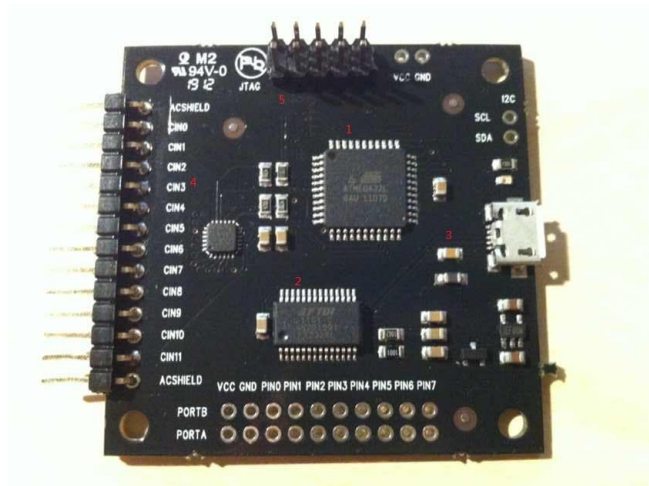


FIGURE 3. Sensor Controller Board

Sensor Controller Board contains five main components.

- (1) Atmel's ATmega 128p 8-bit circuit board is used as a data processor in the PCB. It listens from a serial port commands.
- (2) FT232R is used to convert serial port data to USB format.
- (3) USB port
- (4) Connection pins for a capacitive sensor board
- (5) Connection pins for a programmer.



FIGURE 4.AVR JTAG ICE mk2

A programmer-Download AVR program to Atmel's ATmega 32L 8-bit circuit board. When a user needs to update the software on a microcontroller, JTAG ICE mk2 is the tool for this purpose.

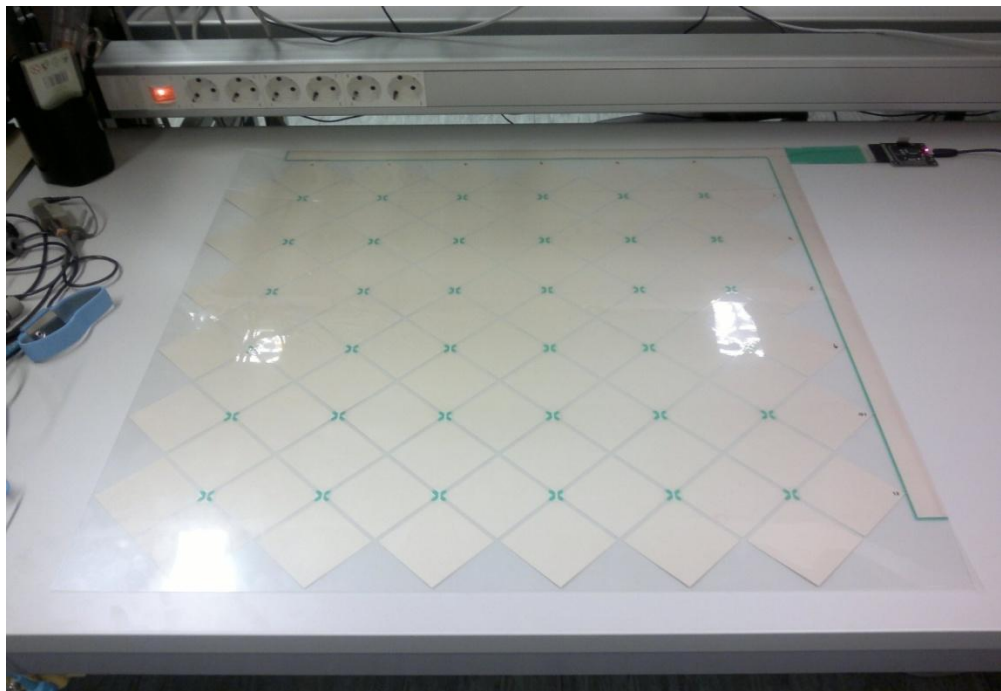


FIGURE 5.Capacitive sensor board

Get the coordinate of user hand moving.

2.2 Requirements



FIGURE 6. Main page of the program

This is the main page, where a museum customer chooses either the grave or the coin hoard.

There are four (4) links or activated areas in this page:

1. The Tervakangas Iron Age burial ground (text & big circle-shaped image),
2. A coin hoard from the Middle Ages in Pattijoki (text & big circle-shaped image)
3. Link “Suomeksi” (in Finnish)
4. Copyright picture at the top right

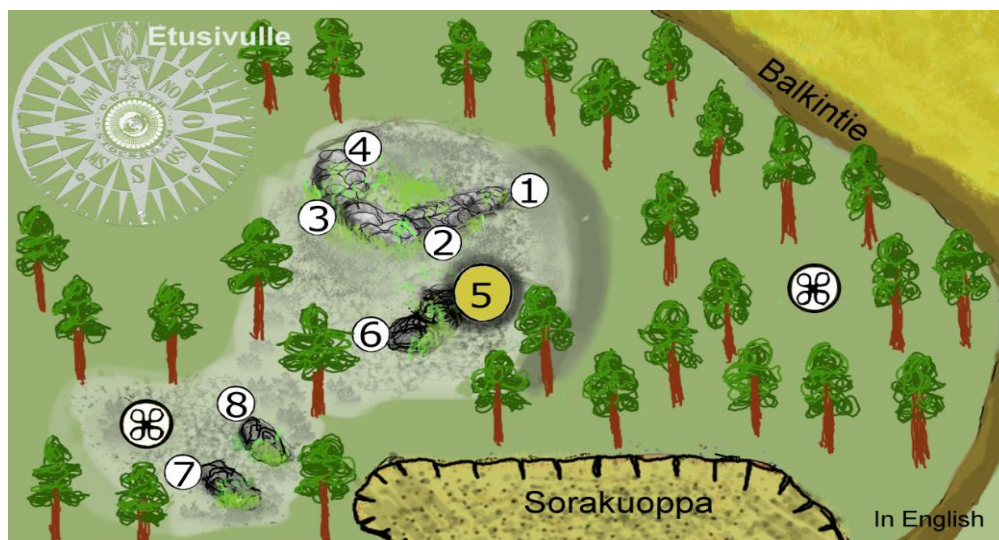


FIGURE 7. Tervakangas Iron Age burial ground

This is the first page of the Tervakangas Iron Age burial ground.

There are 12 activated areas in this page:

1. Ten (10) circle-shaped images (numbers from one to eight and two symbols). Number five is the grave to dig. Other numbers and symbols are about there to find more information, for example in the small information box or on the information page.
2. Link “Suomeksi” (in Finnish)
3. Link “Mainpage” (text & compass image)

Picture ID = 1

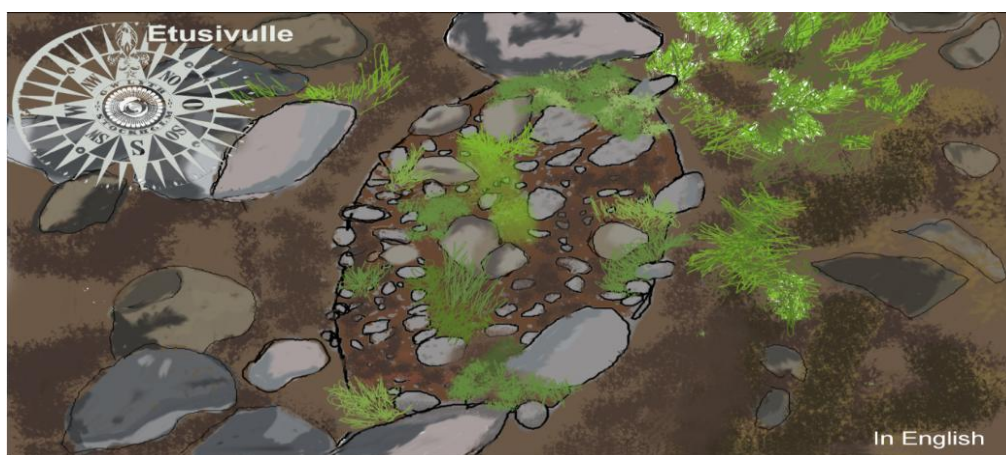


FIGURE 8. The tomb in number 5.

This is the level where the museum customer initiates the grave digging. With the help of hand movements the plants and stones disappear and the color of the country (the earth layers) gradually changes until eventually the grave finds come out.

There are 3 activated areas: 1) link “Mainpage” with a compass image, 2) link “Suomeksi”(in Finnish)and 3) the grave.

Picture ID = 2



FIGURE 9. A broken glass bottle

The broken glass bottle begins to be seen in the grave.

There are 3 activated areas: 1) link “Mainpage” with a compass image, 2) link “Suomeksi” (in Finnish) and 3) the grave.

Picture ID =3



FIGURE 10. Earth layer in grey color.

This earth layer is grey. At this level, a museum customer is using a spatula to dig the grave. You can place the picture of the spatula also in some other sections in this picture. The picture in question is also found separately.

There are 3 activated areas: 1) link “Mainpage” with compass image, 2) link “Suomeksi” (in Finnish) and 3) the grave.

Picture ID = 4

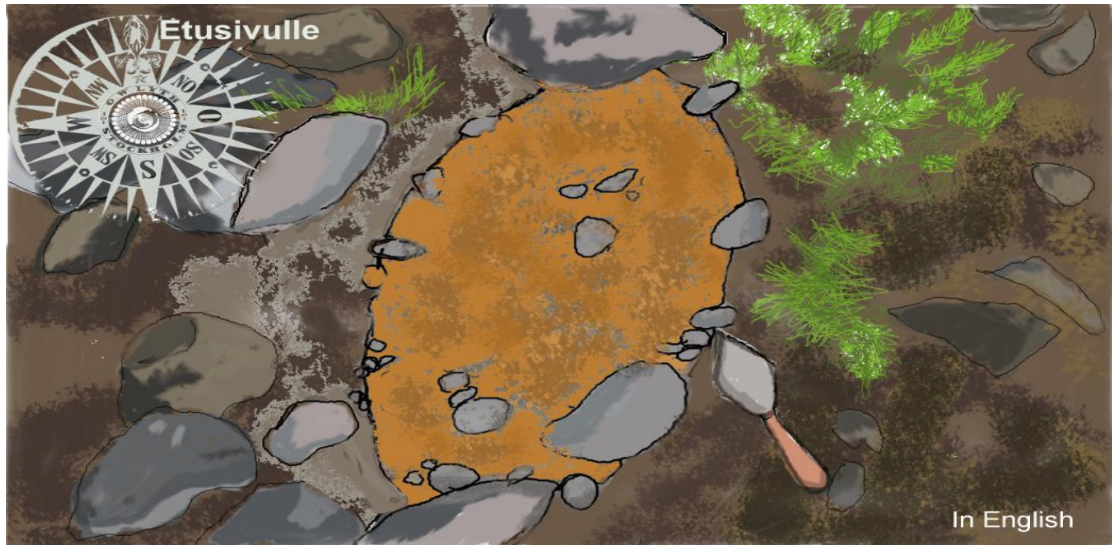


FIGURE 11. Earth layer color changes.

At this level, a museum customer is still using a spatula to dig the grave, but the color changes to a different one.

There are 3 activated areas: 1) link “Mainpage” with compass image, 2) link “Suomeksi” (in Finnish) and 3) the grave.

Picture ID = 5

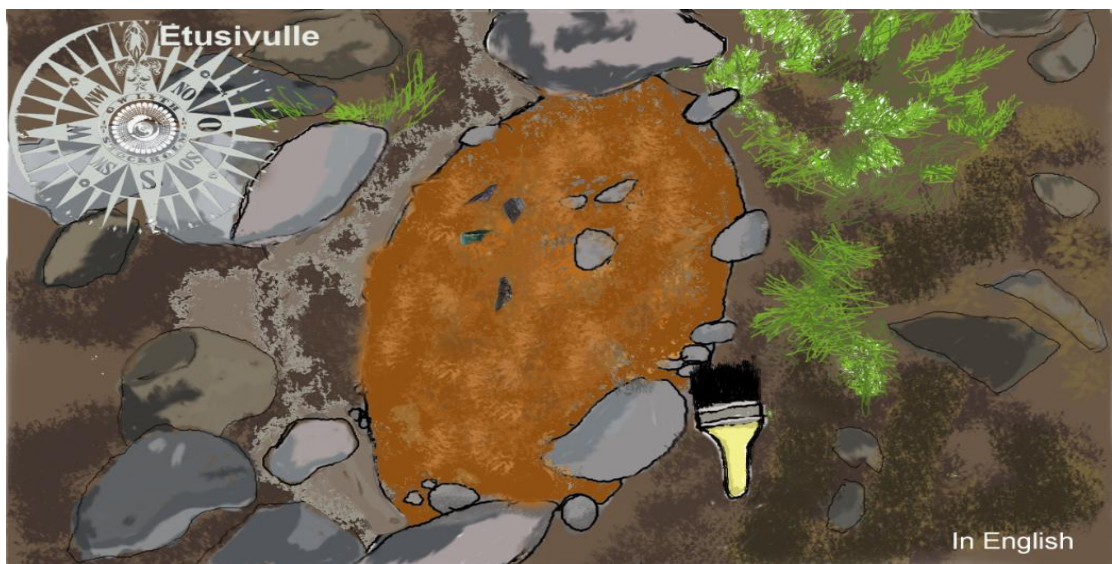


FIGURE 12. Paint brush is used to dig the grave.

At this level, a museum customer is using a paintbrush to dig the grave. You can place the picture of the paintbrush also in some other sections in this picture. The picture in question is also found separately. The color of the grave is still changing to a darker one. The objects of the grave start to show.

There are 3 activated areas: 1) link “Mainpage” with compass image, 2) link “Suomeksi” (in Finnish) and 3) the grave.

Picture ID = 6



FIGURE 13. The color of the grave becomes darker.

At this level, a museum customer is still using a paintbrush to dig the grave and the color of the grave is still changing to a darker one. The objects of the grave start to show more.

There are 3 activated areas: 1) link “Mainpage” with compass image, 2) link “Suomeksi” (in Finnish) and 3) the grave.

Picture ID = 7



FIGURE 14

Museum customer is still using paintbrush to dig grave and the color of the grave is still changing to darker. You can see the objects of the grave and little bit of figure.

There are 3 activated areas: 1) link “Mainpage” with compass image, 2) link “Suomeksi” and 3) the grave.

Picture ID = 8



FIGURE 15

This is a close-up from the grave. When a museum customer points at numbers, he will get additional information about the objects, for example on the separate information page or in the information box.

There are 8 activated areas: 1) link “Mainpage” with compass image, 2) link

“Suomeksi” (in Finnish) and 3) six (6) circle-shaped images (numbers from one to six).

Picture ID = 9



FIGURE 16. Final layer of the grave

This is the final layer of the grave. There is one circle-shaped image. By pointing at that icon, a museum customer will receive additional information about the deceased. There are 3 activated areas: 1) link “Mainpage” with compass image, 2) link “Suomeksi” (in Finnish) and 3) the symbol/icon.

Picture ID = 10

2.3 Important components in communication

USB to serial UART interface (FT232R)

From now on, the preparation has been done. Then we step into the beginning. The connection between the PC and the circuit board is a USB interface. **FT232R** (*It is the latest device to be added to FTDI’s range of USB UART interface Integrated Circuit Devices. The FT232R is a USB to serial UART interface with optional clock generator output, and the new FTDIChip-ID™ security dongle feature. In addition, asynchronous and synchronous bit bang interface modes are available. USB to serial designs using the FT232R have been further simplified by fully integrating the external EEPROM, clock circuit and USB resistors onto the device*). So the connection can be treated as a normal serial port communication.

(Instruction of **FT232R**)

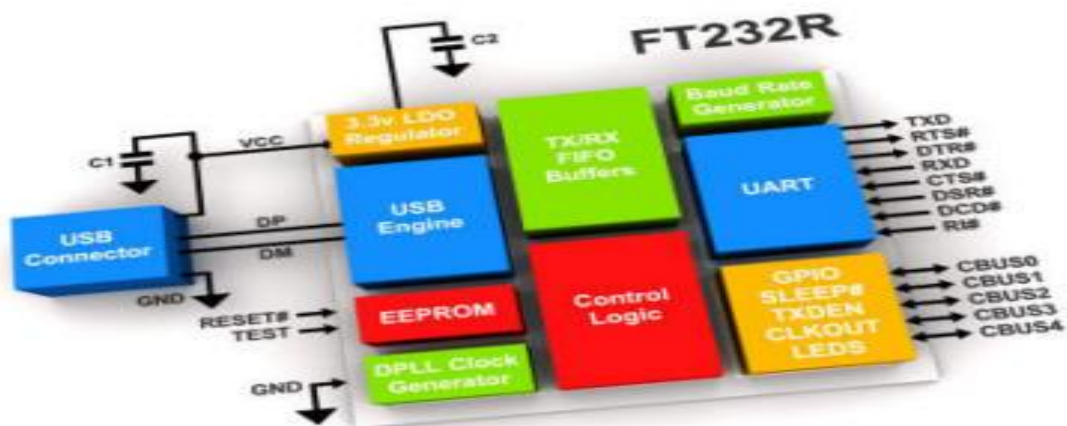


FIGURE 17. FT232R

Capacitance-to-Digital converter (AD7147)

The AD7147 is designed for single electrode capacitance sensors (grounded sensors). AD7147 has an internal algorithm to compensate for environmental changes in capacitance values. However, these are only usable when the decision whether there is a touch detected or not is left for AD7147. In this application where raw capacitance data is read from AD7147 and used for calculations in the microcontroller, this algorithm is not usable. Instead, a user definable threshold value is used in the microcontroller to make the decision.

(Instruction of AD7147)

The flexible sensor board consists of six horizontal lines and six vertical lines with square shaped electrodes. Each line is an independent measurement channel. Capacitance is measured for each channel by AD7147 and is represented by a numerical value between 0 and 65536. Values are calibrated by software to the middle point 32768 in a power up sequence and also by a reset command. The measured values from individual channels can be used to calculate the point of maximum capacitance on board (position of the hand). The board does not support multi touch, meaning that only one maximum capacitance point can be calculated.

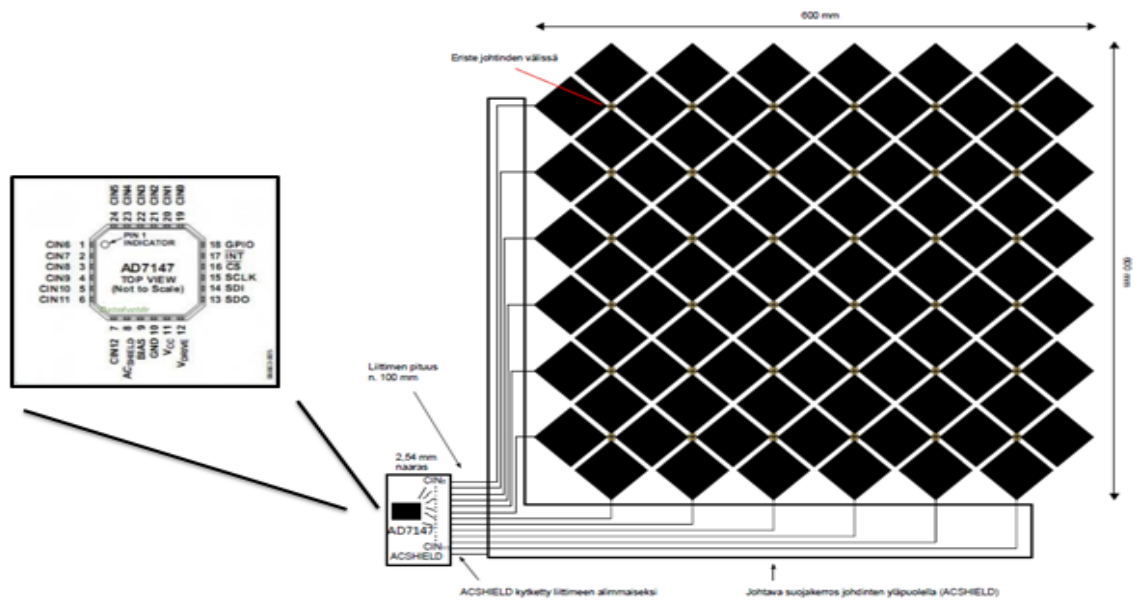


FIGURE 18. Layout of sensor board.

3 Implementation

3.1 Communication

The communication section in this project is based on **MSCOMM Controller** (Designed by Microsoft for a serial port communication). MSComm Controller transfers data through a serial port.

The development IDE is Visual Studio 2010. The Project type is MFC(Dialog)

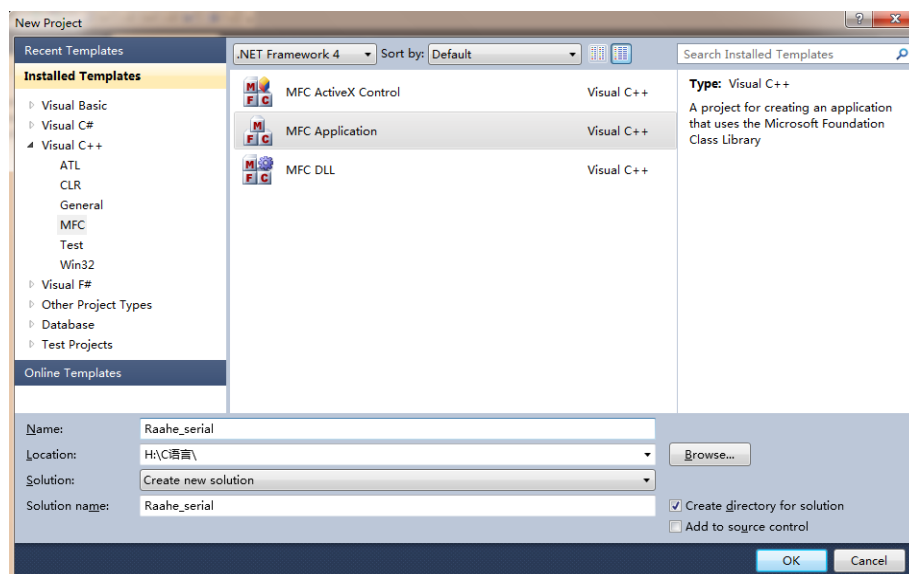


FIGURE 19. Project type

Then drag some components (a Button, an edit control) and the most important thing is to insert an ActiveX control).

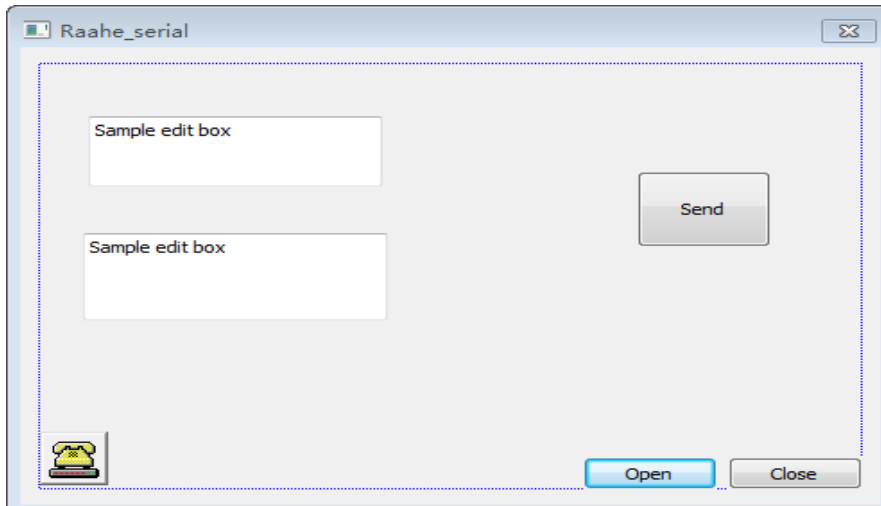


FIGURE20. Example of communication window.

After that, right click the telephone icon on the down left corner to choose a variable, in the dialog window, name a variable and click Finish

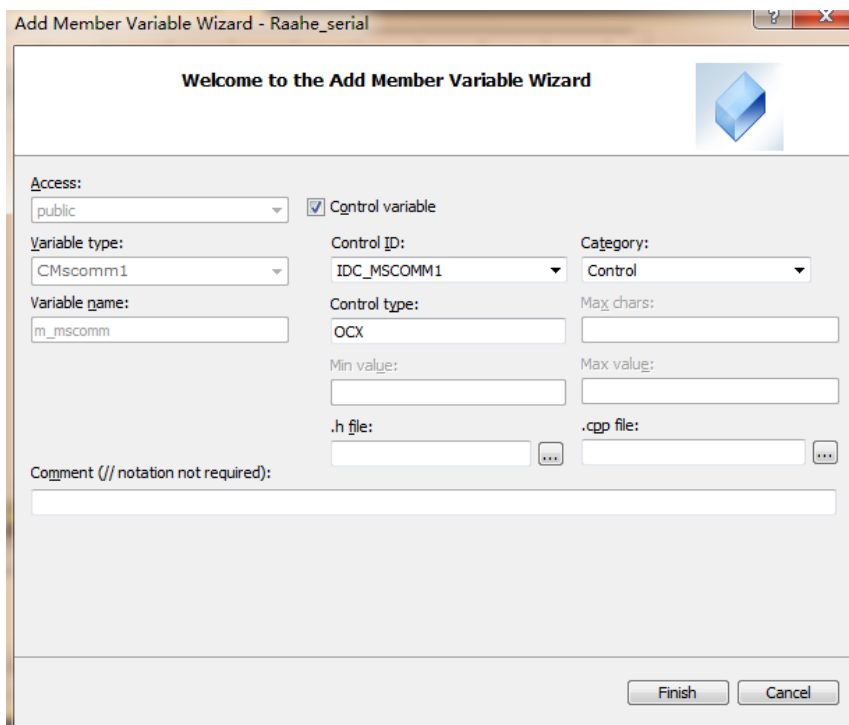


FIGURE 21. Create a communication object.

The preparation seems to be done. Then double click the buttons and enter the code file. Because the sensor board controller can recognize character 'c' as a command which

tells the circuit board to calculate and output a coordinate of the hand above sensor, so the program must implement a real-time sending function (ex. Sending/10ms).

First of all, edit a code of serial initialization in **CTestsunshuo19Dlg::OnInitDialog()**. such as choose a port number, open the serial port, set a baud rate, and data bits, stop a bit and parity and initialize other properties. This action will make sure that serial port works well on transmission.

Character transmission is an essential issue for this project. For serial port communication, a header file should be included using the following code `#include "mscomm1.h"`. mscomm1 as name of the communication object. After that, the whole configuration has been done. Next thing is to code the program. In order to communicate through the serial port, the first step is to keep the port open, as in the initialization paragraph: **CTestsunshuo19Dlg::OnInitDialog()**. The following codes are necessary.

```
if(m_mscomm.get_PortOpen()) // If the port is open. Then chose it
{
    m_mscomm.put_PortOpen(FALSE);
}
m_mscomm.put_CommPort(3); //Choose the port number(In device management)
m_mscomm.put_InBufferSize(1024); //Receiving buffer
m_mscomm.put_OutBufferSize(1024); //Sending buffer
m_mscomm.put_InputLen(0); //Set the length of data to be 0.
m_mscomm.put_InputMode(1); //Read and write data in binary type
m_mscomm.put_RThreshold(1); //When the receiving buffer contains one or more character, program
will execute the communication event(OnComm)
m_mscomm.put_Settings(_T("38400,n,8,1")); //Set the baud rate to be 38400, data bit to be 8 and stop
bit to be 1. Now, the serial port could be opened successfully.
```

Next step is to send a character command to the microcontroller. A function called **send()** is defined in the body of a cpp file. Create a command value : CString command="c";. the send it: m_mscomm.put_Output(COleVariant(command)).

The most important part is the communication event implementation. **void CRAAHEDlg::OnCommMscomm1()**. This event contains a receiving part and a picture switching part.

In the receiving part the received data is added character by character. For example,

```
X: 6 Y: 8 Z: 53725
```

The data shown in the picture is separately received: X : SPACE 6 \r\n Y : SPACE 6 \r\n. Finally got the whole data. The value of X and Y represents the location of a customer's hand above the sensor board.

After that, the program takes out the values of X and Y from the data and it transforms the data type from CString to Integer using atoi(X), integer values are going to be as the parameters in the picture switching function (description later).

Switchpicture(X,Y)

The command sending should be executed several times in a second (ex. 5). So the program needs a timer to do this thing. The following code is for setting a TIMER

```
m_bAutoSend=!m_bAutoSend;
if(m_bAutoSend) {
    SetTimer(1,350,NULL);// The period is350ms
} else {
KillTimer(1); //Cancel TIMER
}
return TRUE; // return TRUE unless you set the focus to a control
}
```

These code is for TIMER implementation after time is up:

```
void CRAAHEDlg::OnTimer(UINT_PTR nIDEvent)
{
    send();// Here the send() is executed 350ms a time
    CDialogEx::OnTimer(nIDEvent);
}
```

3.2 User Interface

After the implementation of a communication program, it is time to do the UI program. It is also based on the window of the communication program, UI diagram as shown below:


```
DrawPicToHDC(image, IDC_STATIC);
```

But if more pictures switching implementation in further coding, these statements must be pasted in every event function. It seems insane, So creating a function called `drawPic(IplImage *image,CString path)`. Set the IplImage object and picture path as parameters.

```
void CshowDlg::drawPic(IplImage *image,CString path){  
    if(image) cvReleaseImage(&image);  
    image = cvLoadImage(path,1);  
    DrawPicToHDC(image, IDC_STATIC);  
}
```

Now it becomes more convenient on coding processing. Then put `drawPic(image,"..\pic\Welcome.jpg");` into **OnPaint()**. After debugging. The window could show picture properly like below:

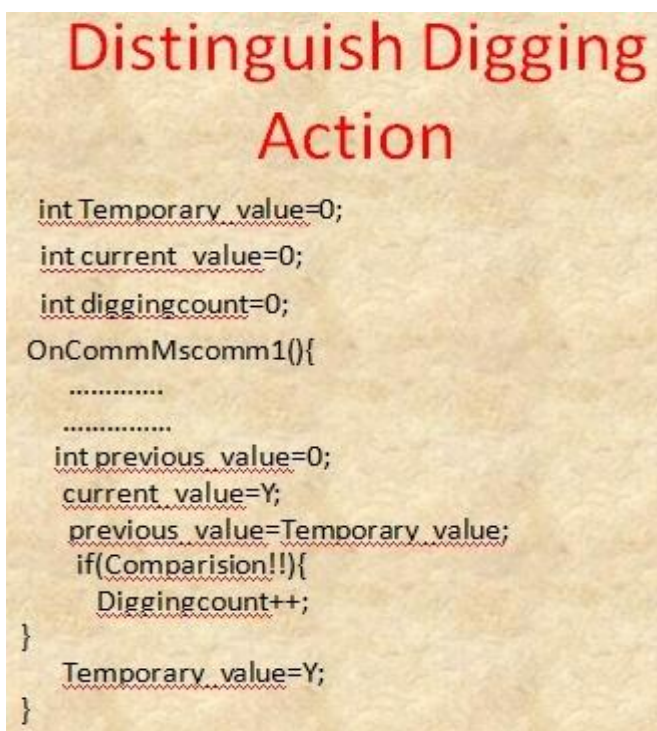


FIGURE 23. Welcoming page

Next thing is to get a picture changed by hand-pointing on the sensor board. The program needs to recognize different areas of hand pointing and tries to change a picture using `drawPic()` in the event handler. But a trouble comes, the second picture just takes a flash and then returns to the welcome picture instantly. That means the program always implements the `drawPic(image,"..\pic\Welcome.jpg");` in **OnPaint()**. For this

case , a `bool` value named `drawable`(default value is `true`) is necessary. After generating the window, the value of `drawable` remains `true`. Once the hand is pointing, the value turns to be `false`. Then the picture control does not show the welcome image any more, instead of the image, you want it to be shown.

An important section is to recognize a digging action (A customer's hand movement from the middle to the side is a digging action). There are three values for this section. One is to store the current coordinate, another one is to store the current coordinate as a temporary value, the third one is to use the temporary value to do a comparison.



```
Distinguish Digging  
Action  
  
int Temporary_value=0;  
int current_value=0;  
int diggingcount=0;  
OnCommMscomm1(){  
.....  
.....  
int previous_value=0;  
current_value=Y;  
previous_value=Temporary_value;  
if(Comparision!!){  
    Diggingcount++;  
}  
    Temporary_value=Y;  
}
```

FIGURE 24. Code section of digging action.

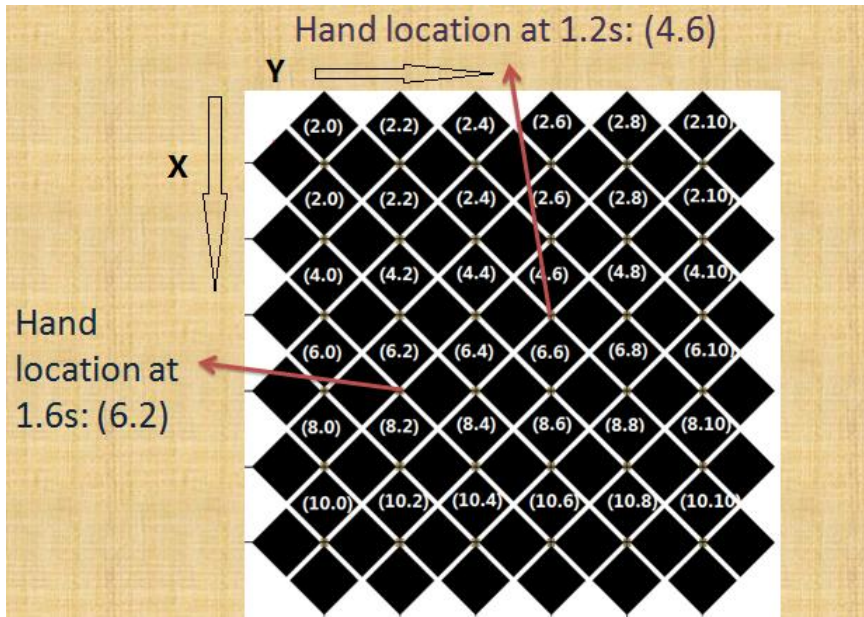


FIGURE 25. Hand location in two different time-point.

4 TESTING

After completing the communication program, connect the controller board with the sensor board and PC do some test by holding a hand above the sensor board.

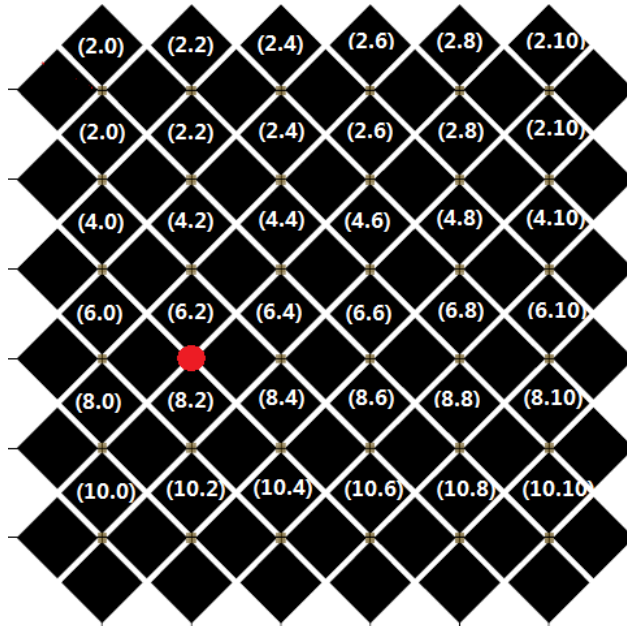


FIGURE 26. Red circle stands the position of hand.

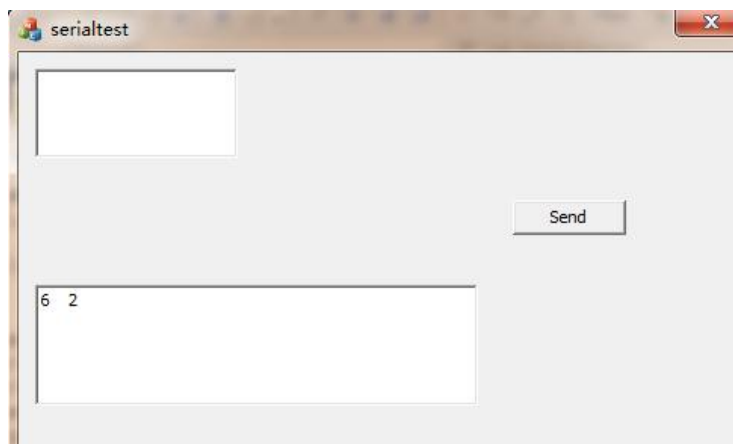


FIGURE 27. Test program made by C++/MFC.

A serial test program is also developed by C++/MFC. The purpose is to make sure a valid communication between the PC and the sensor controller board. Finally, a receiving box shows the hand location.

5 POSSIBILITIES OF FURTHER DEVELOPMENT

5.1 Changing picture

The usage of picture is related to the relative path (ex. "..\\Finnishpic\\sivu 1.jpg"). If a museum person has new pictures and needs to change them in the future, they should just concern the function `drawPic(IplImage *image,CString path)`. which is called in `Switchcase(int _Pic_ID,int X_coordinate,int Y_coordinate)`. Every picture has its own ID number (it is already defined in section 2.2). Make sure which picture you need to have changed and search the ID number in section 2.2. Then switch a new relative path in relating ID case (ex. `case 7`). After that the picture change has been done.

5.2 Changing coordinate recognition

In the UI program, each picture can distinguish its active area by an ID number. So if the picture was changed, may be the active area also changed, and people should measure the coordinate of new active areas. Please follow these steps:

1. Plug the microcontroller board to PC
2. Open HTerm
3. Project the new picture on the curtain
4. Put an object or move your hand above an active area
5. Send a command 'c' to a microcontroller using HTerm
6. Get the measured coordinate

After that, record each measured coordinate of the active area. Finally, switch the old coordinates with the new ones.

5.3 Changing threshold

The value of threshold is a parameter which is related to the sensitivity of the sensor board, the larger the more stable. The current value is almost 500. It means that the sensor board is quite stable and not easily influenced by environmental changes(wind or temperature changes). If you want to change the sensitivity, please follow these steps:

1. Plug the microcontroller board to the PC
2. Open HTerm
3. Send a command 't' to the microcontroller using HTerm(+1)

4. Send a command 'g' to the microcontroller using HTerm(-1)

5.4 Changing music

It is quite easy to change background music or action sounds. Just change the relative path in `PlaySound("relative path",NULL,SND_FILENAME|SND_ASYNC)`.

6 LIST OF REFERENCES

1. Instruction of **FT232R**, Date of retrieval: 04.03.2013

<http://www.ftdichip.com/Products/ICs/FT232R.htm>

2 Instruction of **AD7147**, Date of retrieval: 21.03.2013

<http://wenku.baidu.com/view/f27048da6f1aff00bed51e66.html>

3 Hardware overview, Date of retrieval: 17.02.2013

Spant capacitance sensor board(VTT)

APPENDICES

1 FT232R DRIVER INSTALLATION

The serial-to-usb converter IC used on the sensor hardware circuit board is FTDI's FT232R. A virtual serial port driver needs to be installed before connecting the system to a computer. The driver can be downloaded for various operating systems from IC manufacturer's home page from address: <http://www.ftdichip.com/Drivers/VCP.htm>. The driver installation should be straightforward.

After the driver has been installed successfully the circuit board can be connected to the computer. A red led next to the USB connector indicates that the power is on. Check from Windows' Device manager that a new COM port is visible, titled as "USB Serial Port". Check also the COM port's number (e.g. COM3). The circuit board draws approximately 21 mA current from the USB port.

2 SOFTWARE INTERFACE AND OPERATION

The system is interfaced via a virtual serial port with the following parameters:

- Baud rate: 38400
- Data bits: 8
- Stop bits: 1
- Parity: none

All communication is done using ascii characters, so it is possible to interface to the board using a terminal software, for example Putty or HTerm. Of course, in more advanced applications dedicated software can interface the serial port directly.

In the idle mode, the board is waiting for commands from the computer. A command is essentially a single ascii character. When a command has been received, it is executed and the results are sent back to the computer. Available commands are:

h	Help, list all of commands
a	Reads raw (but calibrated) capacitance values for all channels. The capacitance value for one channel is an integer between 0 and 65536. Each channel is calibrated during power on and reset to 32768. When hand is brought close to the board, the capacitance value increases.
c	Calculates and outputs coordinates of the hand above the sensor. X and Y coordinates (value between 0 and 10) represent the location of the hand and Z coordinate represents the capacitance value (approximately inversely proportional to the distance) in

	that location. Threshold value (see below) is used to decide whether there is a hand above the board. When the threshold is not exceeded, the X and Y coordinate values of 65536 are used to indicate “no touch” situation.
t	Increases the coordinate detection threshold one step. Use this to adjust the sensitivity so that noise does not cause a false detection. The threshold value is stored in a non-volatile memory and is restored during power up. Start with the thresholds between 100 and 150.
g	Decreases the threshold value one step.
r	Resets and recalibrates the sensor. The capacitance value of each channel is adjusted to 32768. Use this when the environment of the board has changed and the capacitance values have drifted away from 32768. This is also done on power up.
o	Same as c, but in a shorter format.

3 KNOWN ISSUES

The sensor board is very sensitive to all capacitance changes around it. It means that when the environment changes, for example the sensor board is moved or a large object is brought near it, the hardware needs to be recalibrated. This can be done in software using the command *r* as mentioned previously.

Another source of possible problems is grounding. Capacitance sensing IC AD7147 measures the capacitance between electrodes on the sensing board and ground net of the board. Because sensing electrodes are relatively far away from the sensing IC, there needs to be a strong ground reference present. Poor ground reference will decrease the sensitivity greatly. In experiments, it has been found out that the system works best in a system where the computer is connected to mains power using a grounded power outlet. This is the optimal situation. In a laptop system running on battery power the sensitivity can be increased by using a long wire between the PC hardware and pc or by soldering an additional wire to the ground net of the hardware board and fixing it in the vicinity of the sensing board

4 RELATED CODE

```
// RAAHEDlg.cpp : implementation file
#include "stdafx.h"
#include "RAAHE.h"
```

```

#include "RAAHEDlg.h"
#include "afxdialogex.h"
#include <Windows.h>
#include <MMSystem.h>
#pragma comment(lib,"WINMM.LIB")
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialogEx
{
public:
    CAboutDlg();

// Dialog Data
    enum { IDD = IDD_ABOUTBOX };

protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support

// Implementation
protected:
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()

// CRAAHEDlg dialog
CRAAHEDlg::CRAAHEDlg(CWnd* pParent /*=NULL*/)
    : CDialogEx(CRAAHEDlg::IDD, pParent)

```

```

{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CRAAHEDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_MSCOMM1, m_mscomm);
}

BEGIN_MESSAGE_MAP(CRAAHEDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_TIMER()
END_MESSAGE_MAP()

// CRAAHEDlg dialog

CRAAHEDlg::CRAAHEDlg(CWnd* pParent /*=NULL*/)
    : CDialogEx(CRAAHEDlg::IDD, pParent)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CRAAHEDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_MSCOMM1, m_mscomm);
}

BEGIN_MESSAGE_MAP(CRAAHEDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_TIMER()
END_MESSAGE_MAP()

```

```

// CRAAHEDlg message handlers
bool m_bAutoSend =false;
BOOL CRAAHEDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // Add "About..." menu item to system menu.

// IDM_ABOUTBOX must be in the system command range.
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);

CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
    BOOL bNameValid;
    CString strAboutMenu;
    bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
    ASSERT(bNameValid);
    if (!strAboutMenu.IsEmpty())
    {
        pSysMenu->AppendMenu(MF_SEPARATOR);
        pSysMenu->AppendMenu(MF_STRING,
IDM_ABOUTBOX, strAboutMenu);
    }
}

// Set the icon for this dialog. The framework does this automatically
// when the application's main window is not a dialog
SetIcon(m_hIcon, TRUE); // Set big icon
SetIcon(m_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here
    openport();
    m_bAutoSend=!m_bAutoSend;
if(m_bAutoSend)
{

SetTimer(1,350,NULL);//period time is 1000ms
}
else

```

```

{
KillTimer(1); //Cancel timer
}

return TRUE; // return TRUE unless you set the focus to a control
}

```

```

void CRAAHEDlg::OnSysCommand(UINT nID, LPARAM lParam)

```

```

{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialogEx::OnSysCommand(nID, lParam);
    }
}

```

// If you add a minimize button to your dialog, you will need the code below
// to draw the icon. For MFC applications using the document/view model,
// this is automatically done for you by the framework.

```

IplImage *image=NULL;

```

```

bool drawable=true;

```

```

void CRAAHEDlg::OnPaint()

```

```

{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

```

```

// Center icon in client rectangle

```

```

int cxIcon = GetSystemMetrics(SM_CXICON);

```

```

int cyIcon = GetSystemMetrics(SM_CYICON);

```

```

CRect rect;

```

```

GetClientRect(&rect);

```

```

int x = (rect.Width() - cxIcon + 1) / 2;

```

```

int y = (rect.Height() - cyIcon + 1) / 2;

```

```

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialogEx::OnPaint() } }
// The system calls this function to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CRAAHEDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

BEGIN_EVENTSINK_MAP(CRAAHEDlg, CDialogEx)
    ON_EVENT(CRAAHEDlg, IDC_MSCOMM1, 1, CRAAHEDlg::OnCommMscmm1,
    VTS_NONE)
END_EVENTSINK_MAP()

int count=1;
int pic=1;
int draw=0;
//CArray <CPoint,CPoint&> Dig_Array;
CString digging ;
int T=0;
int y=0;
bool layeradd=false;
void CRAAHEDlg::OnCommMscmm1()
{

    // TODO: Add your message handler code here
    static unsigned int cnt=0;
    VARIANT variant_inp;
    COleSafeArray safearray_inp;
    long len,k;
    unsigned int data[1024]={0};
    byte rxdata[1024]; //set BYTE array

    CString strtemp;

    CString X;
    CString Y;

```

```

CString number;
CString n;
CString m_EditReceive;
int pointer=0;

if(m_mscomm.get_CommEvent()==2) //when values is 2, it means buffer is not empty
{
    cnt++;
    variant_inp=m_mscomm.get_Input(); //read buffer
    safearray_inp=variant_inp; //value transfer
    len=safearray_inp.GetOneDimSize(); //get length of data
    for(k=0;k<len;k++)
    {
        safearray_inp.GetElement(&k,rxdata+k);
    }
    for(k=0;k<len;k++) //transfer array to CString
    {
        strtemp.Format(_T("%c"),*(rxdata+k));
        m_EditReceive+=strtemp;
        CString temp=_T("\r\n"); //
        m_EditReceive+=temp+"\r\n";
    }
    //PlaySound("..\dumb.wav",NULL,SND_FILENAME|SND_ASYNC);

    int nLen =m_EditReceive.GetLength();

    // n.Format(_T("%ld"),nLen );

    for(int i=0; i<nLen; i++) {

if(m_EditReceive.GetAt(i) >= '0' && m_EditReceive.GetAt(i) <= '9'){
        number += m_EditReceive.GetAt(i);
    }

}

    int numberLen =number.GetLength();
    n.Format(_T("%d"),numberLen );

    if(numberLen==15){
        X="11";
        Y="11";
    }else if(numberLen==7){
        X=number.GetAt(0);

```



```

        Y=number.GetAt(1);
    }else if(numberLen==8){
        if(number.GetAt(0)=='1'&&number.GetAt(1)=='0'){
            X="10";
            Y=number.GetAt(2);
        }else if(number.GetAt(1)=='1'&&number.GetAt(2)=='0'){
            X=number.GetAt(0);
            Y="10";
        }
    }else if(numberLen==9){
        X="10";
        Y="10";
    }
    int X_int=atoi(X);
    int Y_int=atoi(Y);

    if(draw==0){
        drawPic(image,"..\Finnishpic\sivu 1.jpg");
    }
    draw=1;
    if(pic==4||pic==5||pic==6||pic==7||pic==8){

        int y2=0;

        y= Y_int;
        y2=T;
        if(((y==1&&y2==2)|| (y==1&&y2==3)
|| (y==1&&y2==4)|| (y==1&&y2==5) || (y==2&&y2==3)

|| (y==2&&y2==4)|| (y==2&&y2==5)|| (y==3&&y2==4)|| (y==3&&y2==5)|| ((y==10&&y2==9)||
(y==10&&y2==8) || (y==10&&y2==7)|| (y==10&&y2==6) || (y==9&&y2==8)

|| (y==9&&y2==7)|| (y==9&&y2==6)|| (y==8&&y2==7)|| (y==8&&y2==6)|| (y==7&&y2==6) ) ) {

PlaySound("..\dumb.wav",NULL,SND_FILENAME|SND_ASYNC);

        layeradd=true;
        count++;
    }

    T=Y_int;

}

```

```

        Switchcase(pic, X_int, Y_int);
    }
}

void CRAAHEDlg::OnTimer(UINT_PTR nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    send();
    CDialogEx::OnTimer(nIDEvent);
}

void CRAAHEDlg::send()
{
    // TODO: Add your message handler code here and/or call default
    CString command="c";
    m_mscomm.put_Output(COLEVariant(command)); //send data
}

void CRAAHEDlg::openport()
{
    // TODO: Add your message handler code here and/or call default
    if(m_mscomm.get_PortOpen() //
        m_mscomm.put_PortOpen(FALSE);
    }
    m_mscomm.put_CommPort(3); //choose COM number
    m_mscomm.put_InBufferSize(1024); //receiving buffer
    m_mscomm.put_OutBufferSize(1024); //sending buffer
    m_mscomm.put_InputLen(0); //set the length of current receiving buffer to be 0. It means
read all.
    m_mscomm.put_InputMode(1); //read and write data in binary type
    m_mscomm.put_RThreshold(1); //if the receiving buffer has more than one character, then
implement serial port event(OnComm)
    m_mscomm.put_Settings(_T("38400,n,8,1")); //baud rate to be 38400 , data bit to be 8, stop bit to be 1
    if(!m_mscomm.get_PortOpen()) //if serial port is not open, then open it
    {
        m_mscomm.put_PortOpen(TRUE); //open serial port
    } else
    {
        m_mscomm.put_OutBufferCount(0);
    }
}

```

```

        }
    }

void CRAAHEDlg::DrawPicToHDC(IplImage *img, UINT ID)
{
    CDC *pDC = GetDlgItem(ID)->GetDC();
    HDC hDC= pDC->GetSafeHdc();
    CRect rect;
    GetDlgItem(ID)->GetClientRect(&rect);
    CvvImage cimg;
    cimg.CopyOf( img ); // copy image
    cimg.DrawToHDC( hDC, &rect ); // show the picture on the picture control
    ReleaseDC( pDC );
}

void CRAAHEDlg::drawPic(IplImage *image,CString path)
{
    if(image) cvReleaseImage(&image);
    image = cvLoadImage(path,1);
    DrawPicToHDC(image, IDC_STATIC);
    cvReleaseImage(&image); // release memory
}

int layer=0;

void CRAAHEDlg::Switchcase(int _Pic_ID, int X_coordinate,int Y_coordinate)
{
    switch(_Pic_ID)
    {
    case 1:
        if(X_coordinate==4&&Y_coordinate==2){
            //play music
            PlaySound("../dw020.wav",NULL,SND_FILENAME|SND_ASYNC);
            drawPic(image,"../Finnishpic\\s1_akartta sivu1.jpg");
            pic=2;
        }

        break;

    case 2:
        if(X_coordinate==6&&Y_coordinate==4){

            PlaySound(NULL,NULL,NULL); //stop
            music

```

```

drawPic(image, "..\\Finnishpic\\info_argeologinenkaivaus.jpg");
        pic=3;
    }else if(X_coordinate==2&&Y_coordinate==0){
        PlaySound(NULL,NULL,NULL);
        drawPic(image, "..\\Finnishpic\\sivu 1.jpg");
        pic=1;
    }

break;

    case 3:
        if(X_coordinate==10&&Y_coordinate==2){
            //drawing
            drawPic(image, "..\\Finnishpic\\s1_hautapiirros_aloitus.jpg");
                pic=4;
            }else if(X_coordinate==2&&Y_coordinate==0){
                drawPic(image, "..\\Finnishpic\\sivu 1.jpg");
                pic=1;
            }

            break;

            case 4:

                if(layeradd==true){
                    if(count%2==0){
                        layer++;
                    }
                }

                if(layer==1){
                    drawPic(image, "..\\Finnishpic\\s1_hautapiirros_aloitus.jpg");
                    layeradd=false;
                }else if(layer==2){
                    drawPic(image, "..\\Finnishpic\\s2_hautapiirros.jpg");
                    layeradd=false;
                }else if(layer==3){
                    drawPic(image, "..\\Finnishpic\\s3_hautapiirros.jpg");
                    layeradd=false;
                    pic=5;
                    layer=0;
                }

                if(X_coordinate==2&&Y_coordinate==0){
                    drawPic(image, "..\\Finnishpic\\sivu 1.jpg");
                    layeradd=false;
                    layer=0;
                }

```

```

        pic=1;
    }
break;
    case 5:
        if(layeradd==true){
            if(count%2==0){
                layer++;
            }
        }
        if(layer==1){
            drawPic(image,"..\Finnishpic\s4_hautapiirros.jpg");
            layeradd=false;
        }else if(layer==2){
            drawPic(image,"..\Finnishpic\s5_hautapiirros.jpg");
            layeradd=false;
        }else if(layer==3){
            drawPic(image,"..\Finnishpic\s6_pullo_pois.jpg");
            layeradd=false;
        }
        else if(layer==4){
            drawPic(image,"..\Finnishpic\s7_pullo_haipuu_2.jpg");
            layeradd=false;
            pic=6;
            layer=0;
        }

        if(X_coordinate==2&&Y_coordinate==0){
            drawPic(image,"..\Finnishpic\sivu 1.jpg");
            layeradd=false;
            layer=0;
            pic=1;
        }
break;
    case 6:
        if(layeradd==true){
            if(count%2==0){
                layer++;
            }
        }
        if(layer==1){
            drawPic(image,"..\Finnishpic\s8_pullo_haipuu_3.jpg");
            layeradd=false;

```

```

        }else if(layer==2){
drawPic(image,"..\Finnishpic\s9_pullo_haipuu.jpg");
layeradd=false;
        }else if(layer==3){
drawPic(image,"..\Finnishpic\s10_pullo_haipunut.jpg");
layeradd=false;
        }
else if(layer==4){
drawPic(image,"..\Finnishpic\s11_pullo_haipunut.jpg");
layeradd=false;
pic=7;
layer=0;
        }

if(X_coordinate==2&&Y_coordinate==0){
drawPic(image,"..\Finnishpic\sivu 1.jpg");
layeradd=false;
layer=0;
pic=1;
        }

break;

        case 7:

                if(layeradd==true){

                                if(count%2==0){

                                        layer++;

                                }

                }

                if(layer==1){

drawPic(image,"..\Finnishpic\s12_harmaa_oranssiin_1.jpg");
layeradd=false;
                }else if(layer==2){

drawPic(image,"..\Finnishpic\s13_harmaa_oranssiin_2.jpg");
layeradd=false;
                }else if(layer==3){

drawPic(image,"..\Finnishpic\s14_harmaa_oranssiin_3.jpg");
layeradd=false;
                }
                else if(layer==4){

```

```

drawPic(image,"..\Finnishpic\s15_harmaa_oranssiin_valmis.jpg");
    layeradd=false;
    }
    else if(layer==5){

drawPic(image,"..\Finnishpic\s16_oranssista_esineisiin_1.jpg");
    layeradd=false;
    }
    else if(layer==6){

drawPic(image,"..\Finnishpic\s17_oranssista_esineisiin_2.jpg");
    layeradd=false;
    pic=8;
    layer=0;
    }

    if(X_coordinate==2&&Y_coordinate==0){
    drawPic(image,"..\Finnishpic\sivu 1.jpg");
    layeradd=false;
    layer=0;
    pic=1;
    }

    break;

    case 8:
        if(layeradd==true){
            if(count%2==0){
                layer++;
            }
        }
        if(layer==1){

drawPic(image,"..\Finnishpic\s18_oranssista_esineisiin_3.jpg");
    layeradd=false;
    }else if(layer==2){

drawPic(image,"..\Finnishpic\s19_esineet_pilkistaa_kokonaan.jpg");
    layeradd=false;
    }else if(layer==3){

drawPic(image,"..\Finnishpic\s20_esineita_enemman_1.jpg");
    layeradd=false;

```

```

    }
    else if(layer==4){

drawPic(image,"..\Finnishpic\s21_esineita_enemman_2.jpg");
        layeradd=false;
    }
    else if(layer==5){

drawPic(image,"..\Finnishpic\s22_esineita_enemman_3.jpg");
        layeradd=false;
    }
    else if(layer==6){
drawPic(image,"..\Finnishpic\s23_esineita_enemman
valmis.jpg");

        layeradd=false;
    }
    else if(layer==7){
drawPic(image,"..\Finnishpic\s23_esineita_enemman
valmis.jpg");

        layeradd=false;
    }
    else if(layer==8){
drawPic(image,"..\Finnishpic\s24esineet_esiin_1.jpg");
        layeradd=false;
    }
    else if(layer==9){
drawPic(image,"..\Finnishpic\s25hautapiirros.jpg");
        layeradd=false;
    }
    else if(layer==10){
drawPic(image,"..\Finnishpic\s26hautapiirros.jpg");
        layeradd=false;
    }
    else if(layer==11){
drawPic(image,"..\Finnishpic\s27hautapiirros.jpg");
        layeradd=false;
    }
    else if(layer==12){
drawPic(image,"..\Finnishpic\s28hautapiirros.jpg");
        layeradd=false;
    }
    else if(layer==13){

```



```

        drawPic(image, "..\\Finnishpic\\s29_esineet_lahelta.jpg");
        layeradd=false;
        pic=9;
        layer=0;
    }

    if(X_coordinate==2&&Y_coordinate==0){
        drawPic(image, "..\\Finnishpic\\sivu 1.jpg");
        layeradd=false;
        layer=0;
        pic=1;
    }

    break;

    case 9:

        if(layeradd==true){
            if(count%2==0){
                layer++;
            }
        }

        if(X_coordinate==2&&Y_coordinate==0){
            drawPic(image, "..\\Finnishpic\\sivu 1.jpg");
            pic=1;
            layer=0;
        }

        break;
    }
}

bool CRAAHEDlg::diggingAction(CString A )
{

    CString Y1,Y2;
    Y1=A.GetAt(0);
    Y2=A.GetAt(1);
    int x1=atoi(Y1);int x2=atoi(Y2);
    if(x1>x2||x1<x2){
        return true;
    }
    /else{
        return false;
    }}
}

```