Post-laboratory

Recalculation of FF for one type

R	v_i
100	5.566×10^{-4}
200	1.012×10^{-3} ✓
400	6.834×10^{-4}
800	3.72×10^{-4}
1600	1.933×10^{-4}
3200	9.955×10^{-5}
6400	5.83×10^{-6}
9999	3.20×10^{-5}
50	2.952×10^{-4}
25	1.488×10^{-4}
13	7.62×10^{-6}

- Realised I have to take the maximum $v_i = I_m V_m$ to calculate FF and η.

$$Vi_{max} = 1.012 \times 10^{-3} \text{ VA/W}$$

$$FF = \frac{1.012 \times 10^{-3}}{(2.5 \times 10^{-3})(0.55)} = 0.736$$

$$\eta = 0.736 \left(\frac{2.5 \times 10^{-3} \times 0.55}{45 \times 1.96 \times 10^{-6}} \right) = 11.47$$

P20:

25/April/2024

2.1 Read up threads and mutexes

- Continued reading https://hpc-tutorials.llnl.gov/posix/:

Notes: Mutex variable - implement thread synchronization and protects shared data when multiple writes ~~occur~~ occur

- prevent "race" conditions

race condition - occurs when two or more threads can access shared data and they try to change it at the same time

Signed by Originator _____ Print Name _____ Date _____

Signed by Witness _____

some source: https://stackoverflow.com/questions/34510/what-is-a-race-condition

example: if (x == 5) // the "Check"
{
y = x * 2; // The "Act"
// If another thread changed x in between "if (x==5)" and
// "y = x * 2" above,
// y will not be equal to 10.
}

2.2: Read up on using the GPIO interface
- Looked back on C7: Serial Communications to review UART.
- We would use UART - Universal Asynchronous Receiver/Transmitter to send and receive data between two Raspberry Pi.
- Read https://pinout.xyz/pinout/uart for the pins. UART pins in wiringPi are 15, 16.
- Read https://projects.drogon.net/raspberry-pi/wiringpi/serial-library/ to learn about ~~wiringP~~ wiringSerial.h library for serial communication.
- Read https://www.waveshare.com/wiki/Raspberry_Pi_Tutorial_Series:_Serial to read more about wiringSerial.h. (shows example program)

2.3 Design your application

? Send-window: Use mouse clicks as signals for a slot that updates the ~~Paint~~ paintEvent(). This slot takes in x and y inputs and store them as ~~class~~ protected class members. The paintEvent() then gets the x and y inputs from the member to update the window.

? Represent the drawing commands as x and y ~~in~~ values and those x & y are being sent at a time.

Signed by Originator _____ Print Name _____ Date _____

◇ Send x and then send y using void serialPutchar( ) inside the while(1) loop. (optional) logic 0 start bit → data bits → PB → stop bit logic 1

◇◇ - Read https://github.com/WiringPi/WiringPi/blob/master/examples/blink-thread.c to understand wiring threads in wiringPi.h.

PI_THREAD (NAME) // routine that is called when // thread of NAME is created

{

~~rout~~ ...

}

int main (void)

{ :

~~pid~~ piThreadCreate (NAME); // creates thread called NAME

return 0;

}

- Read https://forums.raspberrypi.com/viewtopic.php?t=73501 about locking piLock/piUnlock functions to make threads safe.

piLock (int keyName);

~~piUnlo~~ piUnlock (int keyName);

◇ Use three wires : GND↔GND, TX1 → RX2, TX2 → RX1 . pins 15 & 16.

Note: Use https://abyz.me.uk/rpi/pigpio/pdif2.html to use PiGPIO Library and its serial ~~fn~~ communication codes.

3. Implementation

(3.1) The GUI send and receive windows

- Copied the P6 source/skeleton code to a newly created project file folder P20.

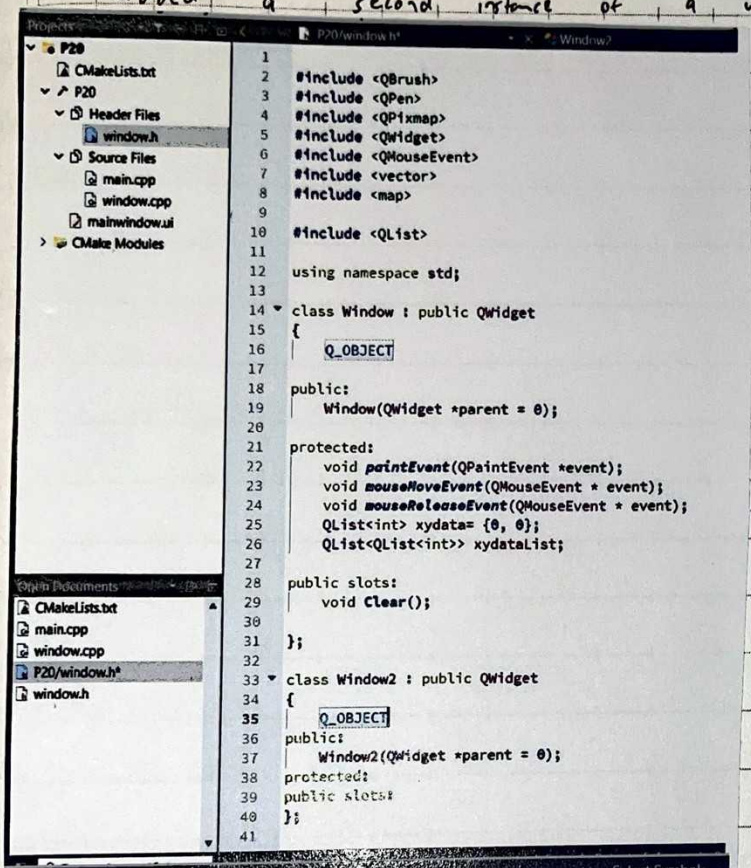
- ~~Mr~~ Added QList<int> xydata = {0, 0}; to store a temporary x & y ~~values~~ value pair.

- Added QList< QList<int>> xydataList to append store the x & y value pair ~~pairs~~ that was ~~were~~ stored in xydata.

Signed by Originator _____ Print Name _____ Date _____

Signed by Witness _____ Print Name _____

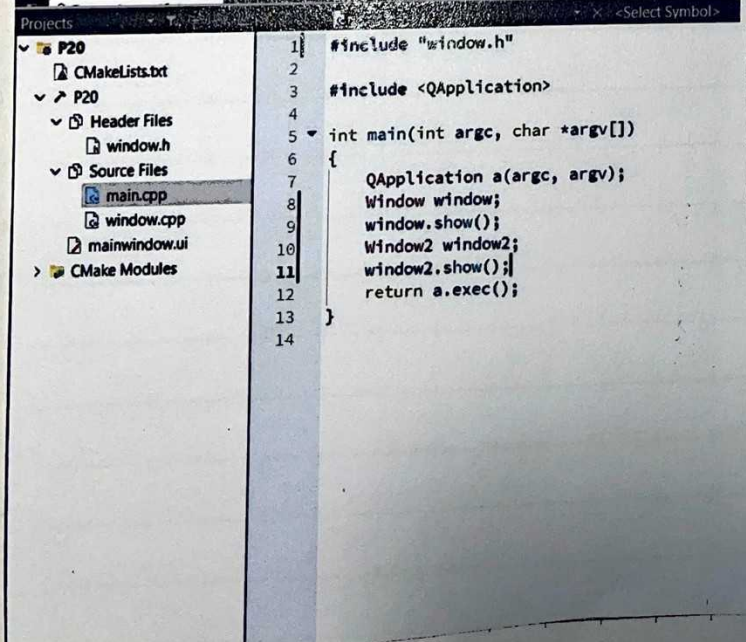
- Modified mouseReleaseEvent and added mouseMoveEvent.
- Added a Clear button to erase the drawing.
- Added a for loop to iterate through xyDataList to draw them as points in response to mouse clicks & holds in paintEvent().
- Added a second instance of a window for the receiving window.



Declaration of class Window and Window2

Window = send-window

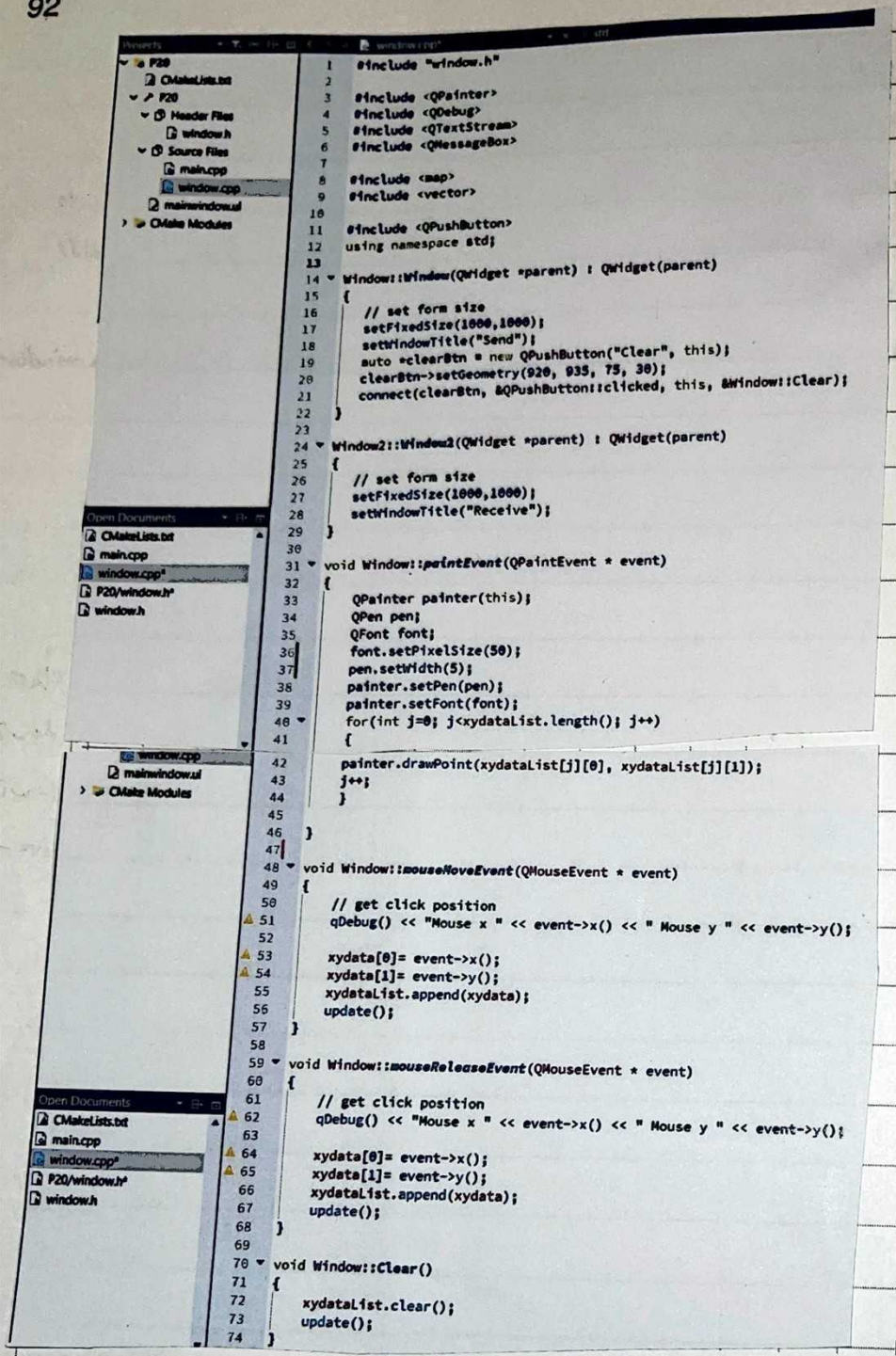
Window 2 = receive-window



main() program

Signed by Originator _____ Print Name _____ Date _____

Signed by Witness _____ Print Name _____ Date _____



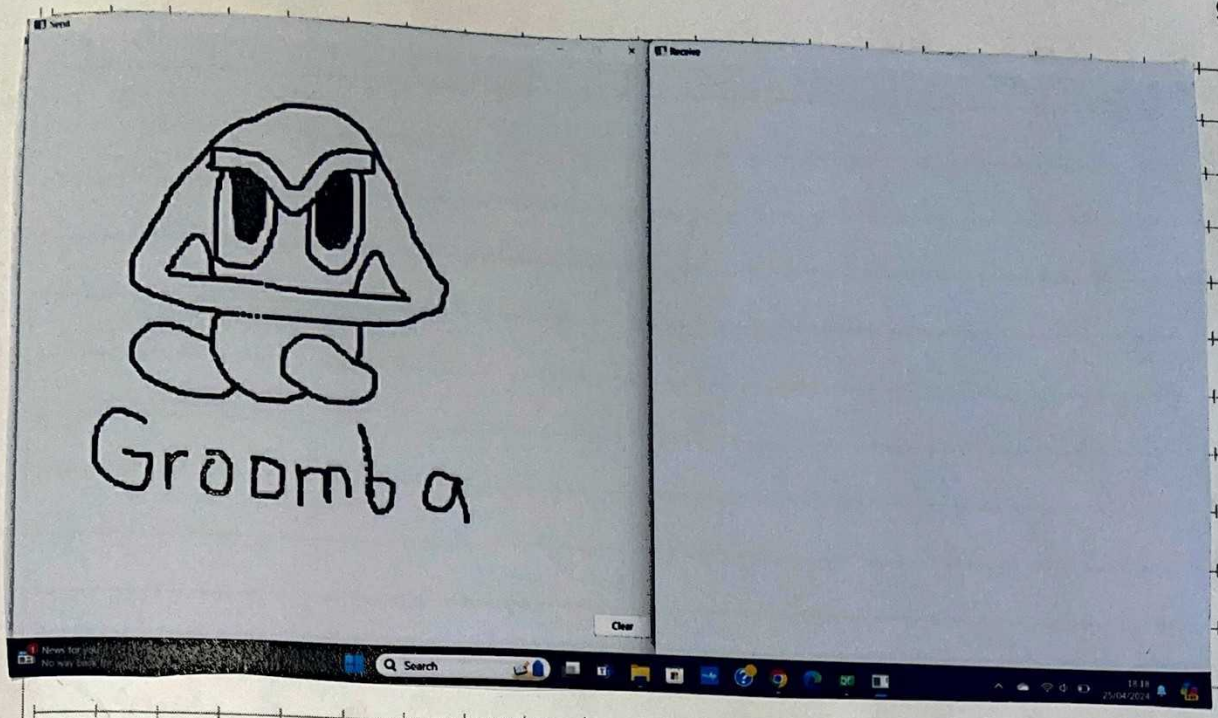
```
#include "window.h"

#include <QPainter>
#include <QDebug>
#include <QTextStream>
#include <QMessageBox>

#include <map>
#include <vector>

#include <QPushButton>
using namespace std;

Window::Window(QWidget *parent) : QWidget(parent)
{
    // set form size
    setFixedSize(1000,1000);
    setWindowTitle("Send");
    auto *clearBtn = new QPushButton("Clear", this);
    clearBtn->setGeometry(920, 935, 75, 30);
    connect(clearBtn, &QPushButton::clicked, this, &Window::Clear);
}

Window2::Window2(QWidget *parent) : QWidget(parent)
{
    // set form size
    setFixedSize(1000,1000);
    setWindowTitle("Receive");
}

void Window::paintEvent(QPaintEvent * event)
{
    QPainter painter(this);
    QPen pen;
    QFont font;
    font.setPixelSize(50);
    pen.setWidth(5);
    painter.setPen(pen);
    painter.setFont(font);
    for(int j=0; j<xydataList.length(); j++)
    {
    painter.drawPoint(xydataList[j][0], xydataList[j][1]);
    j++;
    }

}

void Window::mouseMoveEvent(QMouseEvent * event)
{
    // get click position
    qDebug() << "Mouse x " << event->x() << " Mouse y " << event->y();

    xydata[0]= event->x();
    xydata[1]= event->y();
    xydataList.append(xydata);
    update();
}

void Window::mouseReleaseEvent(QMouseEvent * event)
{
    // get click position
    qDebug() << "Mouse x " << event->x() << " Mouse y " << event->y();

    xydata[0]= event->x();
    xydata[1]= event->y();
    xydataList.append(xydata);
    update();
}

void Window::Clear()
{
    xydataList.clear();
    update();
}
```

Initialisation of Window and Window2 constructors, paintEvent() to draw and update the drawing, for loop inside the paintEvent() to iterate through xydataList to draw the points, mouseMoveEvent() to ~~de~~ capture the x & y points while mouse click hold, ~~and~~ mouseReleaseEvent() to capture the x & y points with one mouse click, and Clear() to clear xydataList to clear the window.

Initialisation of clearBtn (clear button) is inside the Window constructor.

Note: Got the idea of mouseMoveEvent() from MMFid Morsted. Got the idea of ~~pa~~ storing points from Evans Kutiloy ~~Abdulla Randh~~ and Abdulla Moosa. Abdulla Randh Moosa.

Signed by Originator _____ Print Name _____



Graphical User Interface of the program. The left is the send window and the right is the ~~transmitter~~ receive-window.

27/April/2024

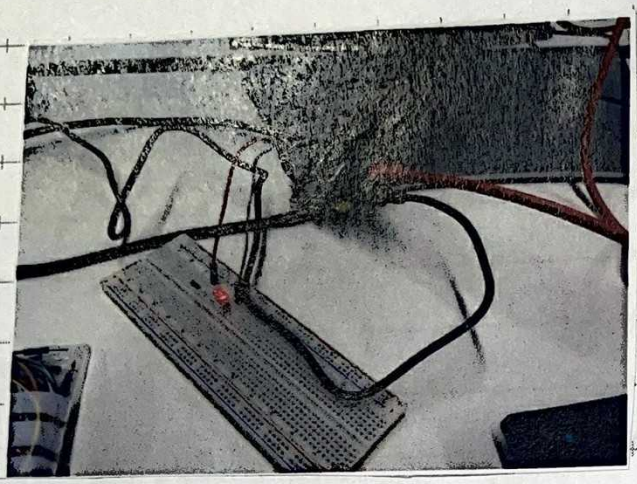
- Went to building 60 ~~of~~ and connected RPi to a computer.
- Followed the instruction of setting up Pi GPIO with Qt in blackboard.
- Read and followed https://abz.me.uk/rpi/pigpio/download.html to download and install the library.
- It didn't work.
- Read and followed https://abz.me.uk/rpi/pigpio/download.html instead.
- Made a ~~file in~~ text file blink.c inside pigpio-master
- connected a LED on gpio 25.

```
#include <pigpio.h>
#include <stdio.h>

int main()
{   if (gpioInitialise()<0){} else{}
    gpioSetMode(25, PI_OUTPUT);
    while (1) {
        gpioWrite(25, 1);
        gpioDelay(1000000);
        gpioWrite(25, 0);
        gpioDelay(1000000);
    }
    gpioTerminate();
    return 0;
}
```

Signed by Originator _____ Print Name _____ Date _____

Signed by Witness _____ Print Name _____ Date _____

- Compiled and ran by typing and entering
gcc -Wall -pthread -o blink blink.c -lpigpio -lrt
sudo ./blink
in the directory of pigpio-master.
- The LED successfully showed blinking.



- Added #include <pigpio.h> to the main.cpp file.
- Added find_library(PIGPIO_LIBRARY pigpio PATHS /usr/local/lib)
target_link_libraries(LED PRIVATE ${PIGPIO_LIBRARY})
at the end of the CMakeLists.txt file.
- Added the blink code to the main() & commented out the window instances and built and ran it to test the GPIO with the Qt.
- ~~for~~ Had an error " cannot specify link libraries for target "LED" which is not built by this project.
- Replaced LED with P20.
- Built and ran the code but it gave gpio Write ~~ini~~initialised and my monitor screen froze.
- Unplugged the power and plugged it again.
- Added ~~init~~ int i = 0; edited while(1) into while(i<10) and added i++; at the end so the program do not run forever.
- Copied the code from https://github.com/joan2937/pigpio/blob/master/util/FindPigpio.cmake and pasted it and replaced

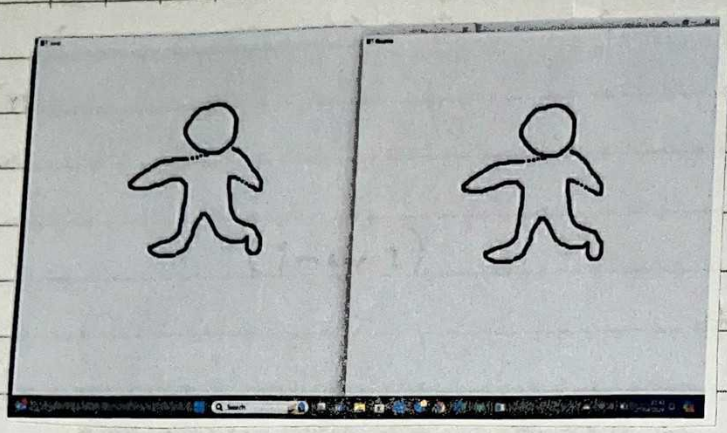
Signed by Originator

the code from the instruction in CMakeLists.txt:
- Debugged the main file and had undefined reference errors.
- Used this code:
find_library (PIGPIO_LIBRARY
NAMES pigpio.h
PATHS /home/theoy/pigpio-master)
target_link_libraries (P20 PRIVATE ${PIGPIO_LIBRARY})
- Built and ran the code with the same error but this time, I noticed I had "Sorry, you don't have permission to run this program".
- Followed step 8 of the instruction.
- Opened the terminal on the P20 directory.
- Typed & entered "sudo killall pigpiod"
- Typed & entered "make all" but it output "make: *** No rule to make target 'all'. Stop.
- Typed and entered "qmake P20.pro"
and "make".
- Had an error: No such file or directory: QWidget
7 | #include <QWidget>
^~~~~~
- ~~Moved~~ Renamed Windows into sendWindow and sinkWindow for clarity.
- Renamed xydata1 and xydataList21 into xydata2 and xydataList2.
- Read https://doc.qt.io/qt-6/signalsandslots.html to learn more about using slots and signals to exchange data between the window.
- Added sinkWindow::receive() slot and void sendWindow::send() signal and connected them in the main().
- Added paintEvent() to sinkWindow to draw xydataList2,
- Added #include <thread>
#include <iostream> in main.cpp.

Signed by Originator _____ Print Name _____ Date _____

Signed by Wit

- Read Used https://cplusplus.com/reference/thread/thread/ for reference on using std::thread.
- Built and ran P20 and successfully showed the correct output: send window sends data to receive window.



(3.2) Serialize and deserialize drawing-commands and receive

Problem: How will I send 16 bits with only 26 GPIO pins? 28/April/2024

- Realized that slots already run in parallel with each other / run concurrently / only depends on signals.

255 ÷ 2	1	127	48 ÷ 2	0	24	LSB
127 ÷ 2	1	63	24 ÷ 2	0	12	
63 ÷ 2	1	31	12 ÷ 2	0	6	
31 ÷ 2	1	15	6 ÷ 2	0	3	
15 ÷ 2	1	7	3 ÷ 2	1	1	
7 ÷ 2	1	3	1 ÷ 2	1	0	MSB
3 ÷ 2	1	1				
1 ÷ 2	1	0				

- Used the code from P2 (dectobin). and the algorithm $\sum_{i=1}^{N} bit_i \times 2^i$ to make sendWindow::serialise(), and sink slots - window::deserialise(). Dectobin was the algorithm above

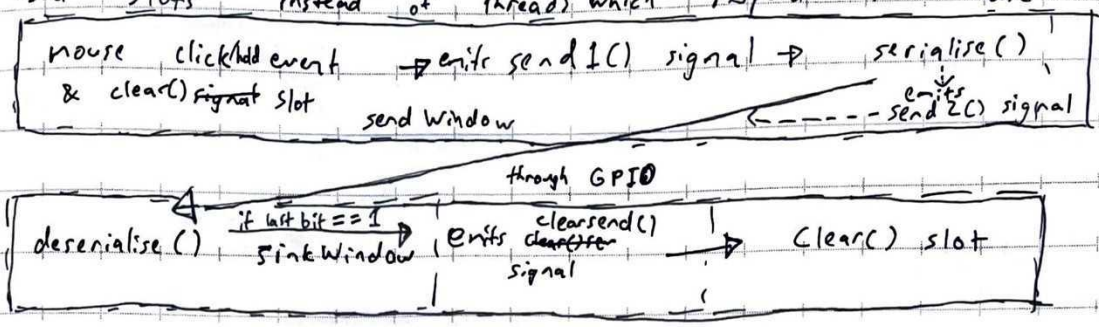
Signed by Originator _____ Print Name _____ Date _____

Signed by Witness _____ Print Name _____ Date _____

https://www.geeksforgeeks.org/program-decimal-binary-conversion.

to convert integer into binary, source:
https://prepinsta.com/cpp-program/cpp-program-for-binary-to-decimal-conversion/

- Used slots instead of threads which respond to mouse clicks/holds.



- Added the clear bit to the serialised bit array to erase the receive window.
- Reduced the window size to 255 x 255 to ~~accomodate~~ limit the x/y values to 8 bits and the bit array to 17 bits (16 xydata bits + 1 cleardata bit).

window.h

Declaration ~~Initialisation~~ of sendWindow and sinkWindow

```
4   #include <QPixmap>
5   #include <QWidget>
6   #include <QMouseEvent>
7   #include <vector>
8   #include <map>
9
10  #include <QList>
11
12  using namespace std;
13
14  class sendWindow : public QWidget
15  {
16      Q_OBJECT
17
18  public:
19      sendWindow(QWidget *parent = 0);
20      QList<int> xydata= {0, 0};
21      QList<QList<int>> xydataList;
22
23  protected:
24      void paintEvent(QPaintEvent *event);
25      void mouseMoveEvent(QMouseEvent * event);
26      void mouseReleaseEvent(QMouseEvent * event);
27
28  public slots:
29      void Clear();
30      void serialise(QList<int> XYDATA, int CLEAR);
31
32  signals:
33      void send1(QList<int> XYDATA, int CLEAR);
34      void send2(QList<int> XYDATA); //acts as transmit to gpio
35
36  };
37
38  class sinkWindow : public QWidget
39  {
40      Q_OBJECT
41
42  public:
43      sinkWindow(QWidget *parent = 0);
```

Signed by Originator _____ Print Name _____ Date _____

Signed by Witness _____ Print Name _____ Date _____

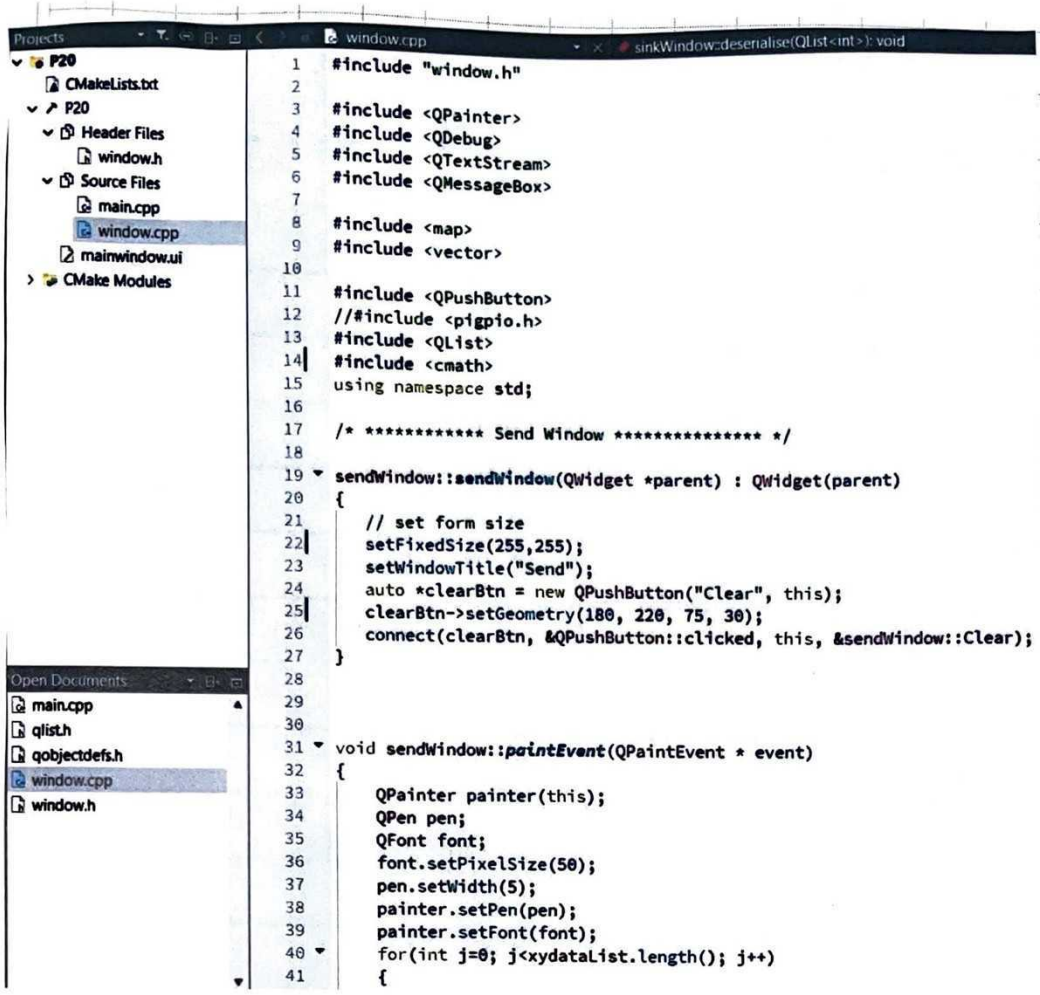
```
    QList<int> xydata2= {0, 0};
    QList<QList<int>> xydataList2;

protected:
    void paintEvent(QPaintEvent *event);

public slots:
    //void receive(QList<int> XYDATA);
    void deserialise(QList<int> XYDATA);
    void Clear();

signals:
    void clearsend();

};
```

Open Documents
- main.cpp
- qlist.h
- qobjectdefs.h
- window.cpp
- window.h

```
#include "window.h"
#include <stdio.h>

#include <QApplication>

#include <thread>
#include <iostream>
#include <QByteArray>
#include <bitset>
#include <vector>
//#include <pigpio.h>

using namespace std;


int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    sendWindow sendwindow;
    sinkWindow sinkwindow;

    QObject::connect(&sendwindow, &sendWindow::send1, &sendwindow, &sendWindow::serialise);
    QObject::connect(&sendwindow, &sendWindow::send2, &sinkwindow, &sinkWindow::deserialise);
    QObject::connect(&sinkwindow, &sinkWindow::clearsend, &sinkwindow, &sinkWindow::Clear);

    sendwindow.show();
    sinkwindow.show();
    return app.exec();
    /*if(gpioInitialise()<0){}
    else{}
    gpioSetMode(25, PI_OUTPUT);
    int i=0;
    while(i<5)
    {
        gpioWrite(25, 1);
        gpioDelay(1000000);
        gpioWrite(25, 0);
        gpioDelay(1000000);
        i++;
```

main.cpp

Instances of sendwindow and sinkwindow of , and connection of slots to signals are shown above.

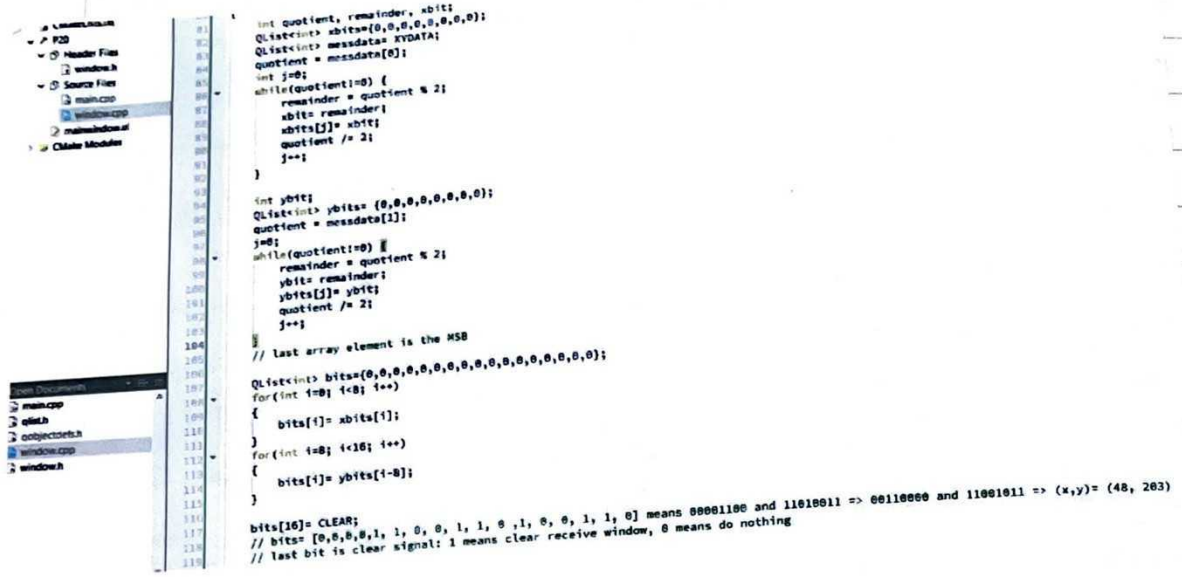



```cpp
#include "window.h"

#include <QPainter>
#include <QDebug>
#include <QTextStream>
#include <QMessageBox>

#include <map>
#include <vector>

#include <QPushButton>
//#include <pigpio.h>
#include <QList>
#include <cmath>
using namespace std;

/* ************ Send Window *************** */

sendWindow::sendWindow(QWidget *parent) : QWidget(parent)
{
    // set form size
    setFixedSize(255,255);
    setWindowTitle("Send");
    auto *clearBtn = new QPushButton("Clear", this);
    clearBtn->setGeometry(180, 220, 75, 30);
    connect(clearBtn, &QPushButton::clicked, this, &sendWindow::Clear);
}


void sendWindow::paintEvent(QPaintEvent * event)
{
    QPainter painter(this);
    QPen pen;
    QFont font;
    font.setPixelSize(50);
    pen.setWidth(5);
    painter.setPen(pen);
    painter.setFont(font);
    for(int j=0; j<xydataList.length(); j++)
    {
    painter.drawPoint(xydataList[j][0], xydataList[j][1]);
    j++;
    }

}
void sendWindow::mouseMoveEvent(QMouseEvent * event)
{
    // get click position
    qDebug() << "Mouse x " << event->x() << " Mouse y " << event->y();

    xydata[0]= event->x();
    xydata[1]= event->y();
    xydataList.append(xydata);
    emit send1(xydata, 0);
    update();
}

void sendWindow::mouseReleaseEvent(QMouseEvent * event)
{
    // get click position
    qDebug() << "Mouse x " << event->x() << " Mouse y " << event->y();

    xydata[0]= event->x();
    xydata[1]= event->y();
    xydataList.append(xydata);
    emit send1(xydata, 0);

    update();
}

void sendWindow::Clear()
{
    xydataList.clear();
    update();
    emit send1(xydata, 1);
}

void sendWindow::serialise(QList<int> XYDATA, int CLEAR)
{
```

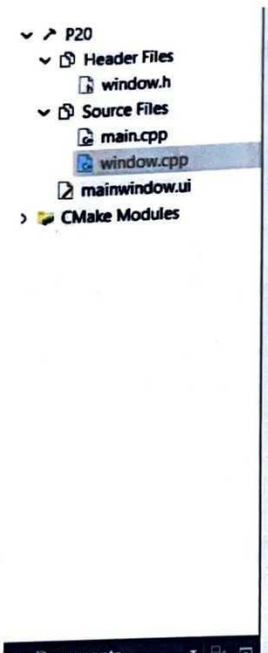
window.cpp

Initialization of the window constructors, events, and slots.

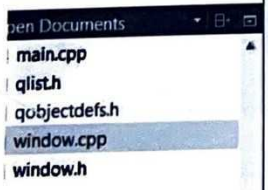
line 79-128: sendWindow::serialise() to serialise xy values and clear signal into a 16 bit ~~bit~~ array of bits to be sent to GPIO.



```
    int quotient, remainder, xbit;
    QList<int> xbits={0,0,0,0,0,0,0,0};
    QList<int> messdata= XYDATA;
    quotient = messdata[0];
    int j=0;
    while(quotient!=0) {
        remainder = quotient % 2;
        xbit= remainder;
        xbits[j]= xbit;
        quotient /= 2;
        j++;
    }

    int ybit;
    QList<int> ybits= {0,0,0,0,0,0,0,0};
    quotient = messdata[1];
    j=0;
    while(quotient!=0) {
        remainder = quotient % 2;
        ybit= remainder;
        ybits[j]= ybit;
        quotient /= 2;
        j++;
    }
    // last array element is the MSB

    QList<int> bits={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    for(int i=0; i<8; i++)
    {
        bits[i]= xbits[i];
    }
    for(int i=8; i<16; i++)
    {
        bits[i]= ybits[i-8];
    }

    bits[16]= CLEAR;
    // bits= [0,0,0,0,1, 1, 0, 0, 1, 1, 0 ,1, 0, 0, 1, 1, 0] means 00001100 and 11010011 => 00110000 and 11001011 => (x,y)= (48, 203)
    // last bit is clear signal: 1 means clear receive window, 0 means do nothing
```

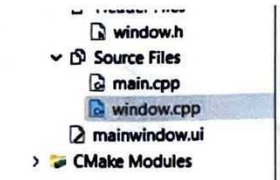


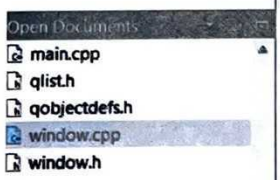
```
    qDebug()<<"start";
    for(int i=0; i<17; i++)
    {
        qDebug()<<"xbits= "<<bits[i];
    }
    qDebug()<<"finish";
    emit send2(bits);
}

/* ************ Receive Window *************** */

sinkWindow::sinkWindow(QWidget *parent) : QWidget(parent)
{
    // set form size
    setFixedSize(255,255);
    setWindowTitle("Receive");
}

void sinkWindow::paintEvent(QPaintEvent * event)
{
    QPainter painter1(this);
    QPen pen;
    QFont font;
    font.setPixelSize(50);
    pen.setWidth(5);
    painter1.setPen(pen);
    painter1.setFont(font);
    for(int j=0; j<xydataList2.length(); j++)
    {
        painter1.drawPoint(xydataList2[j][0], xydataList2[j][1]);
        j++;
    }
}
void sinkWindow::Clear()
{
    xydataList2.clear();
    update();
}
```



line 127 simulates the ~~data~~ array of bits being sent to GPIO.

Signed by Originator _____ Print Name _____ Date _____






```
159
160 void sinkWindow::deserialise(QList<int> XYDATA)
161 {
162     QList<int> encodeddata= XYDATA; //encoded data from gpio input
163
164     QList<int> xbits={0,0,0,0,0,0,0,0}, ybits={0,0,0,0,0,0,0,0};
165
166     for(int i=0; i<8; i++)
167     {
168         xbits[i]= encodeddata[i];
169     }
170     for(int i=8; i<16; i++)
171     {
172         ybits[i-8]= encodeddata[i];
173     }
174
175     int xvalue=0, yvalue=0;
176     for(int i=0; i<8; i++)
177     {
178         xvalue+= xbits[i]*pow(2, i);
179     }
180     for(int i=0; i<8; i++)
181     {
182         yvalue+= ybits[i]*pow(2, i);
183     }
184
185     xydata2[0]= xvalue;
186     xydata2[1]= yvalue;
187     xydataList2.append(xydata2);
188     if(encodeddata[16]==1){emit clearsend();}
189     else{}
190
191     update();
192 }
193
```

```
0,0,0,0,0,0}, ybits={0,0,0,0,0,0,0,0};

ata[i];

)

ddata[i];

;

pow(2, i);

pow(2, i);
```

GUI of the program

sinkWindow:: deserialise() deserialises the array of bits into xy values and the clear signal.

The deserialise() slot triggering from the send2() signal simulates the GPIO.

(3.3) Implement send- and receive-threads 29/April/2024

- Made started making serialise and deserialse threads in main.cpp using std:: threads.

Problem: Had trouble synchronising the threads.

Idea: Use signals.

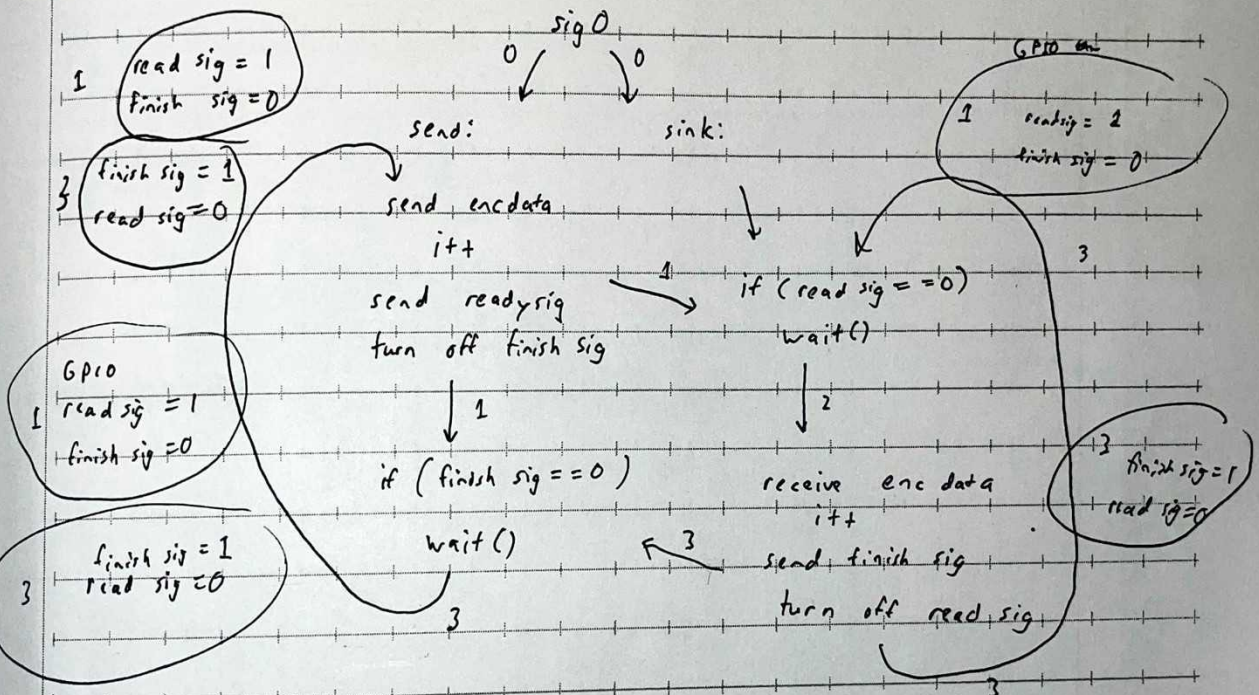
p20 (Continued) 30/April/2024

(3.3) Implement send- and receive threads

~~Continued making the s~~

- Completed making the serialise and deserialise threads that simulate sending encoded data bits one at a time and synchronising them with "ready" and "finish" signal bits.
- If uses std:: threads in main.cpp.

diagram for that made me solve the problem:



(1) Receive-window waits for ready bit before it does anything

(2) Send-window sends data bit (enc data 0) and ready bit (readsig) to receive-window.

(3) Send-window waits for finish bit (finish sig) before it does anything further after (2).

(4) Receive-window receives ready bit, then process the data bit in response, and sends finish bit to send-window

(5) Send-window receives finish bit and starts again at (1) with a new data.

Signed by Originator _____ Print Name _____ Date _____

Signed by Witness _____ Print Name _____ Date _____

Projects

- P20
  - CMakeLists.txt
  - P20
    - Header Files
      - window.h
    - Source Files
      - main.cpp
      - window.cpp
    - mainwindow.ui
  - CMake Modules

Open Documents

- main.cpp
- window.cpp
- window.h

main.cpp

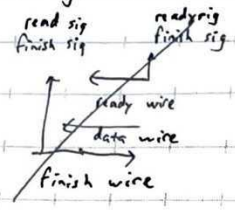
```
#include "window.h"
#include <stdio.h>

#include <QApplication>

#include <iostream>
#include <QByteArray>
#include <bitset>
#include <vector>
#include <thread>
//#include <pigpio.h>
#include <QDebug>
#include <atomic>

using namespace std;
struct {int encdata0; int readysig=0; int finishsig=0;} dataStruct;

/*
(1) Receive-window waits for ready bit before it does anything.
(2) Send-window sends data bit(encdata0) and ready bit (readsig) to receive-window.
(3) Send-window waits for finish bit (finishsig) before it does anything further after (2).
(4) Receive-window receives ready bit, then process the data bit in response, and sends finish bit to send-window.
(5) Send-window receives finish bit and starts again at (1) with a new data.
Process loops at 1->5.
*/

void serialise(atomic<bool>& program_is_running, sendWindow *window);
void deserialise(atomic<bool>& program_is_running, sinkWindow *window);

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    sendWindow sendwindow;
    sinkWindow sinkwindow;

    atomic<bool> running { true } ;

    //connects the slots and signals
    QObject::connect(&sinkwindow, &sinkWindow::clearsend, &sinkwindow, &sinkWindow::Clear);

    //create the threads
    thread serialiseThread(serialise, ref(running), &sendwindow);
    thread deserialiseThread(deserialise, ref(running), &sinkwindow);

    sendwindow.show();
    sinkwindow.show();

    return app.exec();

    //terminate the threads;
    serialiseThread.join();
    deserialiseThread.join();


    /*if(gpioInitialise()<0){}
    else{}
    gpioSetMode(25, PI_OUTPUT);
    int i=0;
    while(i<5)
    {
        gpioWrite(25, 1);
        gpioDelay(1000000);
        gpioWrite(25, 0);
        gpioDelay(1000000);
        i++;

    }
    gpioTerminate();*/

}
//serialise xy values into encoded data to be sent to gpio output
void serialise(atomic<bool>& program_is_running, sendWindow *window)
{
    while(program_is_running)
    {
        int quotient, remainder, xbit;
        QList<int> xbits={0,0,0,0,0,0,0,0};
        QList<int> messdata= window->xydata;
        quotient = messdata[0];
        int j=0;
        while(quotient!=0) {
```

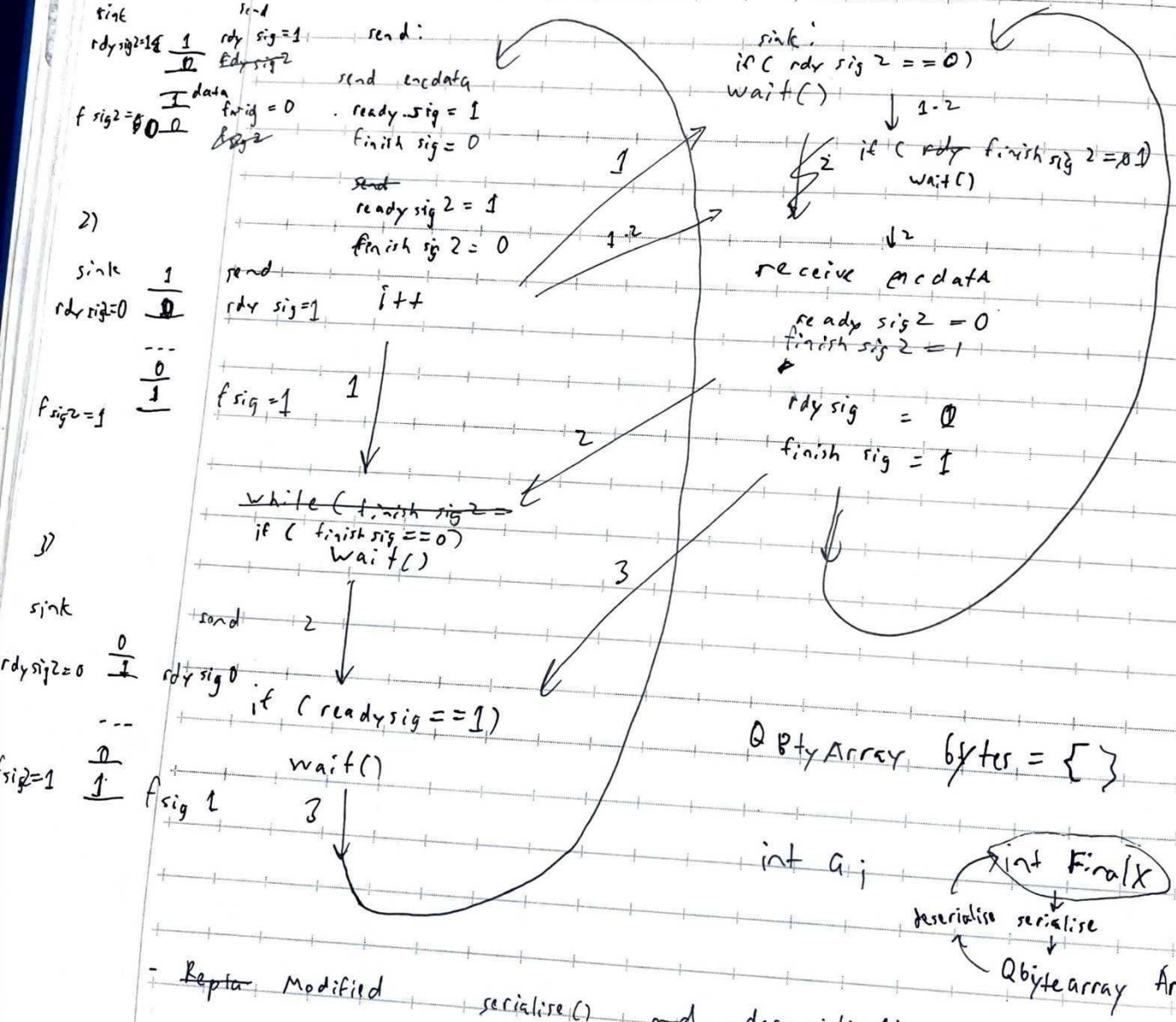
Signed by Originator

```
        remainder = quotient % 2;
        xbits= remainder;
        xbits[j]= xbit;
        quotient /= 2;
        j++;
    }

    int ybit;
    QList<int> ybits= {0,0,0,0,0,0,0,0};
    quotient = measdata[1];
    j=0;
    while(quotient!=0) {
        remainder = quotient % 2;
        ybits= remainder;
        ybits[j]= ybit;
        quotient /= 2;
        j++;
    }
    // last array element is the MSB

    QList<int> bits={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    for(int i=0; i<8; i++)
    {
        bits[i]= xbits[i];
    }
    for(int i=8; i<16; i++)
    {
        bits[i]= ybits[i-8];
    }

    bits[16]= window->clear;
    // bits= {0,0,0,0,1, 1, 0, 0, 1, 1, 0 ,1, 0, 0, 1, 1, 0} means 00001100 and 11010011 => 00110000 and 11001011 => (x,y)= (48, 203)
    // last bit is clear signal: 1 means clear receive window, 0 means do nothing

    //send data to gpio ouput
    int i=0;
    dataStruct.finishsig=0;
    dataStruct.readysig=0;
    int l=1;
    qDebug()<<"encstart";
    while(l==1)
    {
        dataStruct.encdata0= bits[i];
        dataStruct.readysig=1;
        dataStruct.finishsig=0;
        qDebug()<<i<<", "<<dataStruct.encdata0;
        qDebug()<<"send: send readysig";
        i++;
        while(dataStruct.finishsig==0){qDebug()<<"send: wait for finishsig";}
        if(i==17){l=0;}
    }

    qDebug()<<"encfinish";
}
}
//deserialise the encoded binary data from gpio input into xy values
void deserialise(atomic<bool>& program_is_running, sink*window *window)
{

    while(program_is_running)
    {
        qDebug()<<"encdata in deserialise"<<dataStruct.encdata0;
        QList<int> encodeddata= {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; //encoded data from gpio input
        int i=0;
        int l=1;
        qDebug()<<"decstart";
        //receive data from gpio input
        while(l==1)
        {
            while(!dataStruct.readysig==0){qDebug()<<"sink: wait for readysig";}
            encodeddata[i]= dataStruct.encdata0;
            dataStruct.finishsig=1;
            dataStruct.readysig=0;
            qDebug()<<"de_enc"<<i<<","<<dataStruct.encdata0;
            qDebug()<<"sink: send finishsig";
            i++;
            if(i==17){l=0;}
        }
        qDebug()<<"decfinish";

        QList<int> xbits={0,0,0,0,0,0,0,0}, ybits={0,0,0,0,0,0,0,0};

        for(int i=0; i<8; i++)
        {
            xbits[i]= encodeddata[i];
        }
        for(int i=8; i<16; i++)
        {
            ybits[i-8]= encodeddata[i];
        }

        int xvalue=0, yvalue=0;
        for(int i=0; i<8; i++)
        {
            xvalue+= xbits[i]*pow(2, i);
        }
        for(int i=0; i<8; i++)
        {
            yvalue+= ybits[i]*pow(2, i);
        }

        //stores xy value into xydata of receive window
        if(((xvalue==0) && (yvalue==0))==0)
        {
            window->xydata2[0]= xvalue;
            window->xydata2[1]= yvalue;
            window->xydataList2.append(window->xydata2);
        } else{}

        if(encodeddata[16]==1){emit window->clearsend();} else{}    //if last bit==1 (clear signal) then erase receive window

        window->update();
        qDebug()<< "xvalue= "<<xvalue<<", "<<"yvalue= "<<yvalue;
    }

}
```

(3.4) Implement your communication protocol 01 / May / 2024

- Abdulla Raudh Moosa helped me configure the CMake file for wiringPi.h using booleans
- Used wiringPi.h instead of pigpio.h.





- Modified serialise() and deserialise() threads with GPIO functions.
- Helped Lewis H with serialise and deserialise. Guided him on how to research them.

Signed by Originator

Print Name

Date

Signed by Witness

Print Name

sink send

rd.sig = 1 / 1

rdy sig=1

Send:

Send encdata

readysig = 1

digital write (0, 1)

sink:

if (readysig == 0)

wait()

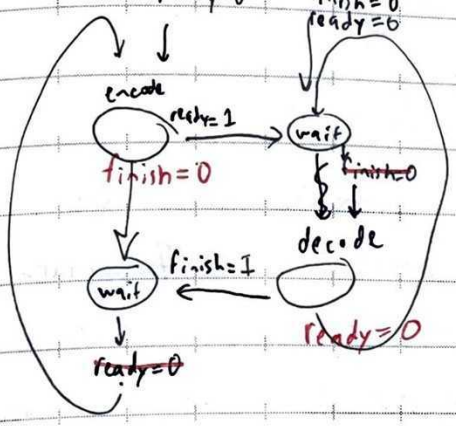
if sig = 1 / 1 t sig

finish sig=0

ready=0

finish=0

ready=0



- Ran and built the program. It successfully writes and reads from GPIO pins but the threads did not synchronize.

01/May/2024

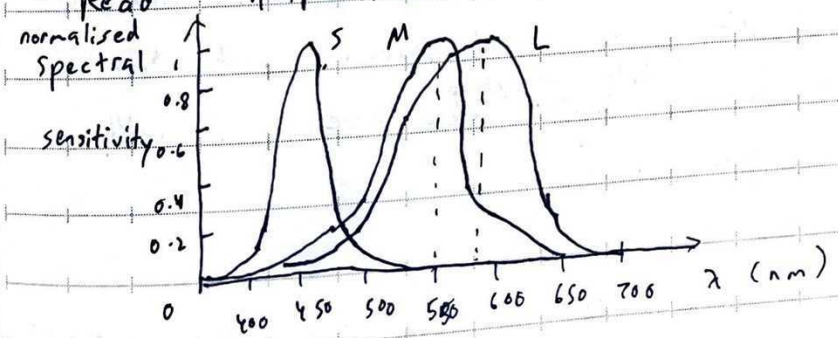
T10: Semiconductor

- Read 1 Aims, Learning Outcomes and Outline section of the laboratory document.

2. Preparation

2.1. The Nature of Colour

- Read https://en.wikipedia.org/wiki/CIE_1931_color_space.



S - short

M - medium

L - long

✓

Tristimulus X, Y, Z

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1.91020 & -1.11212 & 0.20191 \\ 0.37095 & 0.62905 & 0 \\ 0 & 0 & 1.00000 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$ HPE

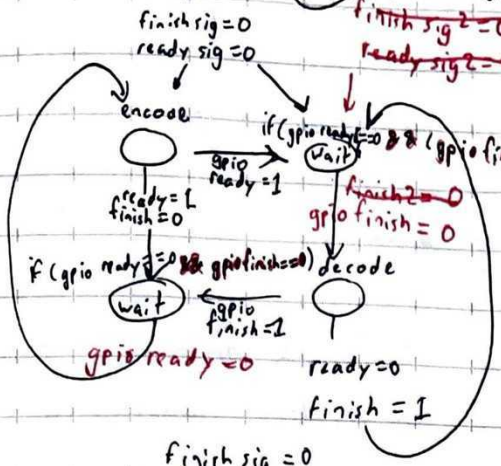
p20 (continued..)

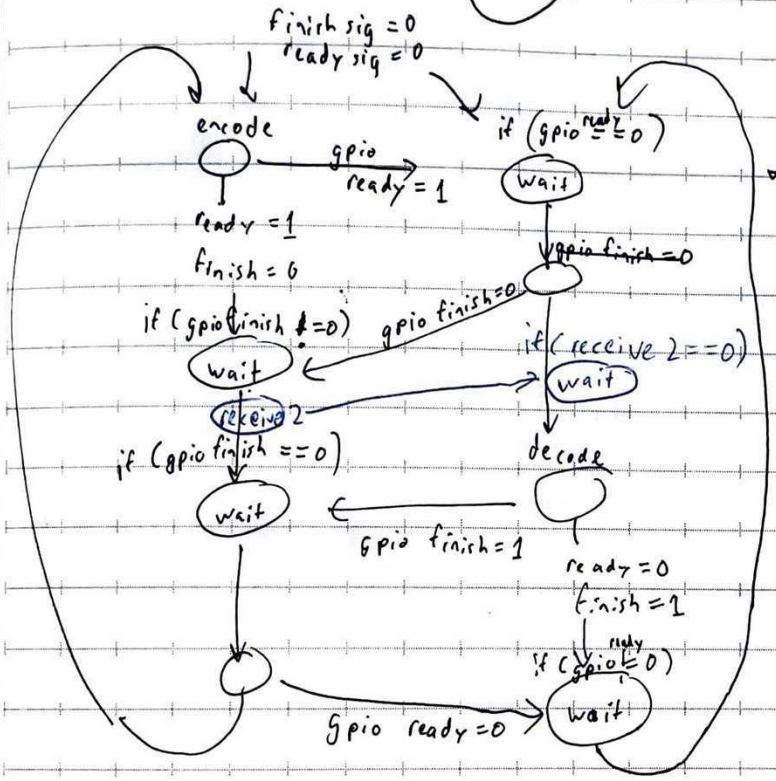
(3.5) Read and write to physical GPIO pins 02/May/2024

~~finish sig 2 = 0~~
~~ready sig 2 = 0~~

without GPIO

~~without~~ with GPIO



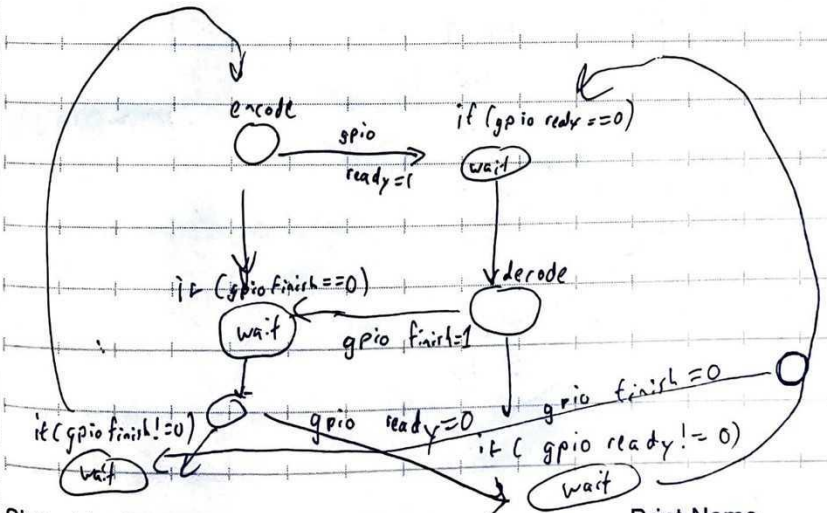


new implementation

~~with GPIO~~ with GPIO

Problem: this creates race conditions

03/May/2024



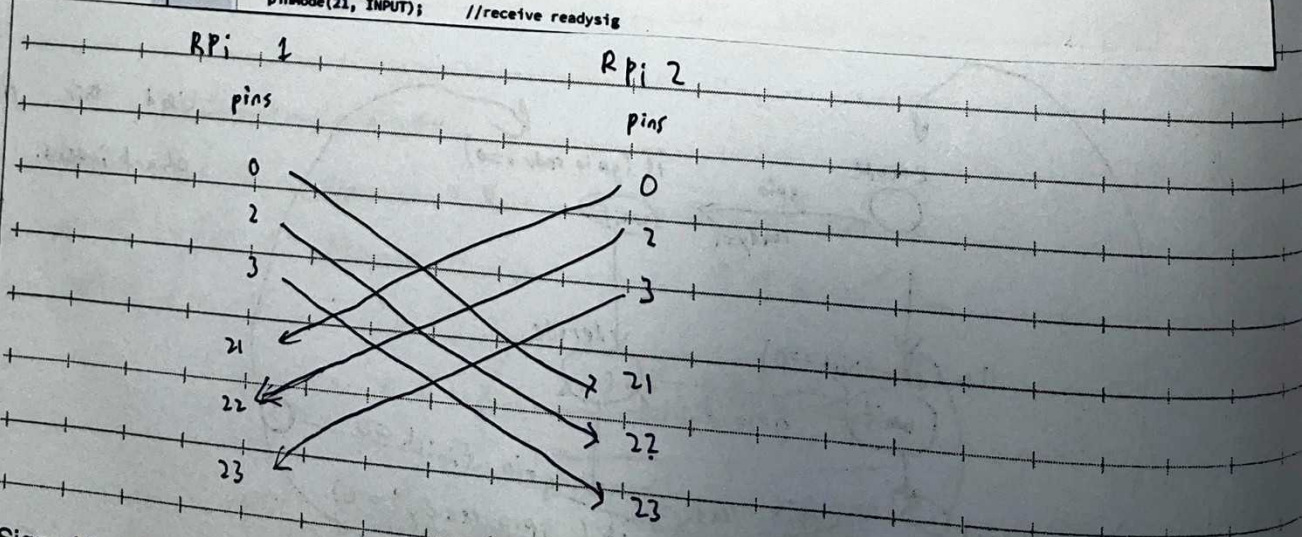
- Used this flow chart instead.

Signed by Originator ..... Print Name ..... Date

Signed by Witness ..... Print Name ..... Date

- Successfully synchronised the receive and send windows using GPIO pins.
- Connected my laboratory partner's raspberry pi with my raspberry pi.
- It successfully sent data between two raspberry pis.
- If one program is run first, the program's send window and the other program's sink window works but not the other way around.

```
1  #include "window.h"
2  #include <stdio.h>
3
4  #include <QApplication>
5
6  #include <iostream>
7  #include <QByteArray>
8  #include <bitset>
9  #include <vector>
10 #include <thread>
11 #include <wiringPi.h>
12 #include <QDebug>
13 #include <atomic>
14 #include <cmath>
15
16 using namespace std;
17
18 /*
19 (1) Receive-window waits for ready bit before it does anything.
20 (2) Send-window sends data bit(encdata0) and ready bit (readsig) to receive-window.
21 (3) Send-window waits for finish bit (finishsig) before it does anything further after (2).
22 (4) Receive-window receives ready bit, then process the data bit in response, and sends finish bit to send-window.
23 Process loops at 1->5.
24 */
25
26
27 void serialise(atomic<bool>& program_is_running, sendWindow *window);
28 void deserialise(atomic<bool>& program_is_running, sinkWindow *window);
29
30 int main(int argc, char *argv[])
31 {
32     wiringPiSetup();
33     //send
34     pinMode(0, OUTPUT);      //send readysig
35
36     pinMode(2, OUTPUT);     //send data
37
38     pinMode(3, INPUT);       //receive finishsig
39
40     //receive
41     pinMode(21, INPUT);      //receive readysig
```



Signed by Originator _____

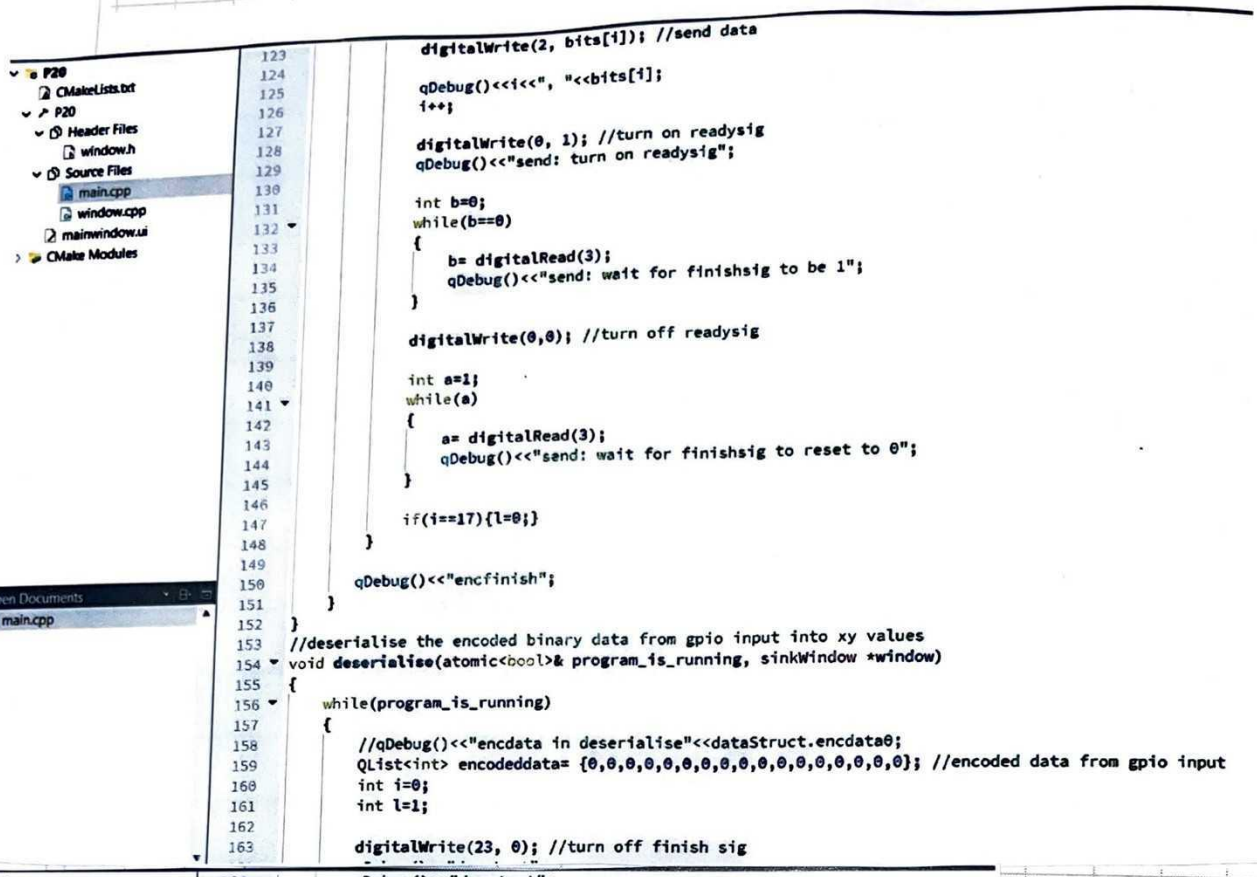
Signed by Witness _____ Print Name

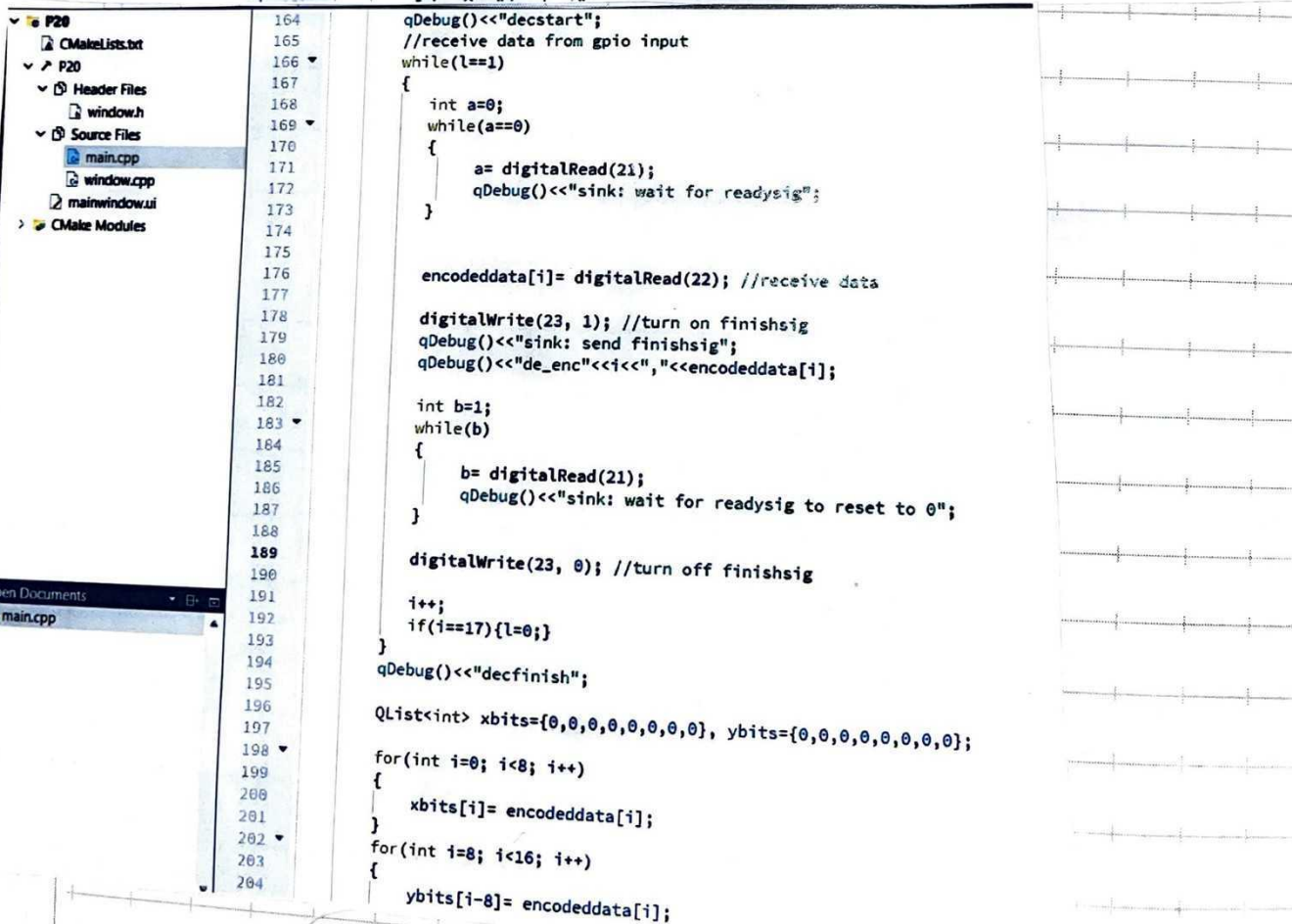
```
    pinMode(22, INPUT);     //receive data

    pinMode(23, OUTPUT);    //send finishsig

    QApplication app(argc, argv);
    sendWindow sendwindow;
    sinkWindow sinkwindow;

    atomic<bool> running { true } ;

    //connects the slots and signals
    QObject::connect(&sinkwindow, &sinkWindow::clearsend, &sinkwindow, &sinkWindow::Clear);

    //create the threads
    thread serialiseThread(serialise, ref(running), &sendwindow);
    thread deserialiseThread(deserialise, ref(running), &sinkwindow);

    sendwindow.show();
    sinkwindow.show();

    return app.exec();

    //terminate the threads;
    serialiseThread.join();
    deserialiseThread.join();

}
//serialise xy values into encoded data to be sent to gpio output
void serialise(atomic<bool>& program_is_running, sendWindow *window)
{
    while(program_is_running)
    {
        int quotient, remainder, xbit;
        QList<int> xbits={0,0,0,0,0,0,0,0};
        QList<int> messdata= window->xydata;
        quotient = messdata[0];
        int j=0;
        while(quotient!=0) {
            remainder = quotient % 2;
```

```
        xbits =remainder;
        xbits[j]= xbit;
        quotient /= 2;
        j++;
    }

    int ybit;
    QList<int> ybits= {0,0,0,0,0,0,0,0};
    quotient = messdata[1];
    j=0;
    while(quotient!=0) {
        remainder = quotient % 2;
        ybit= remainder;
        ybits[j]= ybit;
        quotient /= 2;
        j++;
    }
    // last array element is the MSB

    QList<int> bits={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    for(int i=0; i<8; i++)
    {
        bits[i]= xbits[i];
    }
    for(int i=8; i<16; i++)
    {
        bits[i]= ybits[i-8];
    }

    bits[16]= window->clear;
    // bits= [0,0,0,0,1, 1, 0, 0, 1, 1, 0 ,1, 0, 0, 1, 1, 0] means 00001100 and 11010011 => 00110000 and 11001011 => (x,y)= (48, 203)
    // last bit is clear signal: 1 means clear receive window, 0 means do nothing

    //send data to gpio ouput
    int i=0;
    digitalWrite(0, 0); //turn off ready sig
    int l=1;
    qDebug()<<"encstart";
    while(l==1)
    {
        //dataStruct.encdata0= bits[i];
```

Signed by Originator Print Name Date

Print Name Date


P20
CMakeLists.txt
P20
Header Files
window.h
Source Files
main.cpp
window.cpp
mainwindow.ui
CMake Modules
Open Documents
main.cpp
digitalWrite(2, bits[i]); //send data
qDebug()<<i<<", "<<bits[i];
i++;
digitalWrite(0, 1); //turn on readysig
qDebug()<<"send: turn on readysig";
int b=0;
while(b==0)
{
b= digitalRead(3);
qDebug()<<"send: wait for finishsig to be 1";
}
digitalWrite(0,0); //turn off readysig
int a=1;
while(a)
{
a= digitalRead(3);
qDebug()<<"send: wait for finishsig to reset to 0";
}
if(i==17){l=0;}
}
qDebug()<<"encfinish";
}
}
//deserialise the encoded binary data from gpio input into xy values
void deserialise(atomic<bool>& program_is_running, sinkWindow *window)
{
while(program_is_running)
{
//qDebug()<<"encdata in deserialise"<<dataStruct.encdata0;
QList<int> encodeddata= {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; //encoded data from gpio input
int i=0;
int l=1;
digitalWrite(23, 0); //turn off finish sig


P20
CMakeLists.txt
P20
Header Files
window.h
Source Files
main.cpp
window.cpp
mainwindow.ui
CMake Modules
Open Documents
main.cpp
qDebug()<<"decstart";
//receive data from gpio input
while(l==1)
{
int a=0;
while(a==0)
{
a= digitalRead(21);
qDebug()<<"sink: wait for readysig";
}
encodeddata[i]= digitalRead(22); //receive data
digitalWrite(23, 1); //turn on finishsig
qDebug()<<"sink: send finishsig";
qDebug()<<"de_enc"<<i<<","<<encodeddata[i];
int b=1;
while(b)
{
b= digitalRead(21);
qDebug()<<"sink: wait for readysig to reset to 0";
}
digitalWrite(23, 0); //turn off finishsig
i++;
if(i==17){l=0;}
}
qDebug()<<"decfinish";
QList<int> xbits={0,0,0,0,0,0,0,0}, ybits={0,0,0,0,0,0,0,0};
for(int i=0; i<8; i++)
{
xbits[i]= encodeddata[i];
}
for(int i=8; i<16; i++)
{
ybits[i-8]= encodeddata[i];

- Added destructors for the threads.
- Added a for loop to limit the xy points to be appended to xy points list such that the xy points are within the window.

07/May/2024

- Added a delay using std::chrono to stop the random crashes.
- Made the code neater. source: https://cplusplus.com/forum/general/190880/
- still had the problem of scaling for the second ~~pair that~~ pair in the send of send and receive windows where it drew ~~second~~ after the first pair.

08/May/2024

- Possible solution: use a scaling factor on the deserialised xy of the first pair (the program that ran first and drew).

4 5 5 7 6 90%.

3.1 3.2 3.3 3.4 3.5

Signed by Originator Print Name Date

Print Name Date