【作業内容】

I2Cインターフェースを付加したLCDをラズパイに接続して、文字列が表示できるようにプログラムを作成してください。

【作業項目】

　　1. I2Cインターフェースの準備

　　　i2cdetect -y 1 というコマンドを用いて、I2Cデバイスの番号を調べる。結果、0x27であることがわかった。

　　2. 下調べ

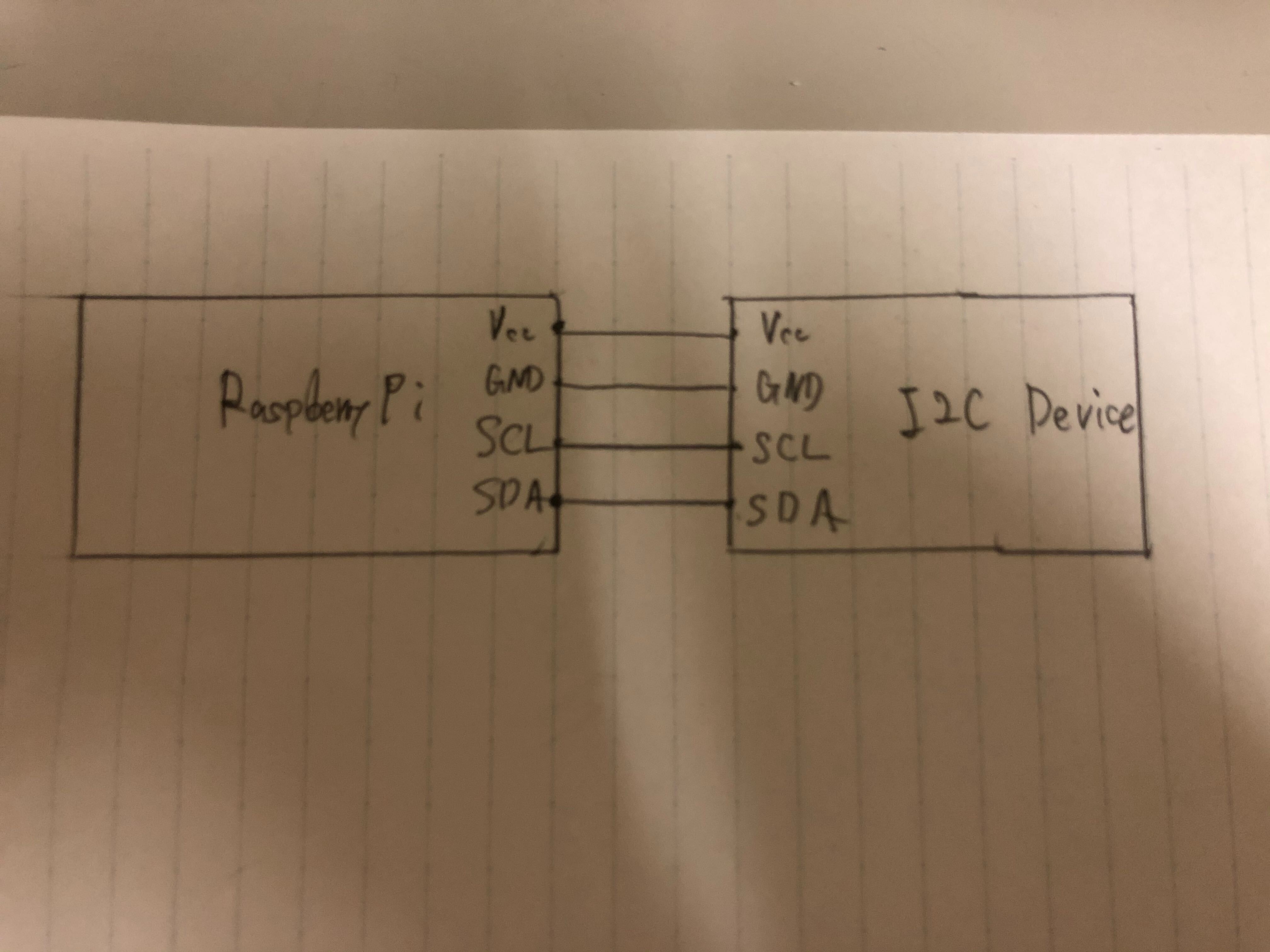
　　　I2Cは通信機器がマスターとスレーブになって通信をする。

　 SCLはクロック信号、SDLはデータ信号を通信しており、SCLがHighのときのSDLの変化をみて、通信の開始と終了を調べている。

　 具体的には、SCLがHighかつSDLが立ち下がると通信開始。通信中に同じことが起こると通信終了となる。したがって、SDLでデータを通信するときは、SDLがLowのときのみ通信する。

　　3. 回路図

　　　RaspberryPiの各ポートとI2Cデバイスを接続する。回路図を以下に示す。



【

　　4. 任意の位置に文字を表示するのは時間の関係上できなかったため、先頭にAを表示するプログラムを示す。ただし、プログラム長が長いため、最後に乗せる。

作業時間】

・作業時間：200分

・報告書作成時間：20分

#include<wiringPi.h>

#include<wiringPiI2C.h>

#include<stdio.h>

#include<stdlib.h>

#define I2CReadMode 1

#define I2CWriteMode 0

/\*

typedef struct {

unsigned int bus4: 1;

unsigned int bus5: 1;

unsigned int bus6: 1;

unsigned int bus7: 1;

} I2CDataBus;

\*/

typedef struct {

unsigned int register\_selector: 1;

unsigned int read\_write: 1;

unsigned int enable: 1;

unsigned int back\_light: 1;

unsigned int bus4: 1;

unsigned int bus5: 1;

unsigned int bus6: 1;

unsigned int bus7: 1;

} I2C8bitDataField;

typedef union {

I2C8bitDataField field;

u\_int8\_t raw\_data;

} I2C8bitData;

I2C8bitData make\_data\_from\_raw(u\_int8\_t data) {

I2C8bitData ret = {.raw\_data = data };

return ret;

}

void show\_i2c\_data(I2C8bitData data) {

printf("[DEBUG] I2C Data Details. %d\n", data.raw\_data);

if (data.field.register\_selector == 1) {

printf(" RS: Command Mode\n");

} else {

printf(" RS: Data Mode\n");

}

if (data.field.read\_write == 1) {

printf(" RW: Read Mode\n");

} else {

printf(" RW: Write Mode\n");

}

if (data.field.enable == 1) {

printf(" Enable Bit: Enable\n");

} else {

printf(" Enable Bit: Disable\n");

}

if (data.field.back\_light == 1) {

printf(" Back Light: On\n");

} else {

printf(" Back Light: Off\n");

}

const u\_int8\_t transmission\_data = (data.raw\_data & 0xF0) >> 4;

printf(" Transmission Data: %d, %d%d%d%d\n", transmission\_data, (transmission\_data & 0b1000) >> 3, (transmission\_data & 0b0100) >> 2, (transmission\_data & 0b0010) >> 1, transmission\_data & 0b0001);

}

u\_int32\_t raw\_I2C\_data\_write(const u\_int32\_t fd, const I2C8bitData data) {

const u\_int32\_t ret = wiringPiI2CWrite(fd, data.raw\_data | 0b1000);

if(ret != 0) {

printf("[ERROR] I2C write error. fd = %d, data = %d", fd, data);

return ret;

}

return 0;

}

void I2C\_data\_write(const u\_int32\_t fd, I2C8bitData data) {

// show\_i2c\_data(data);

data.field.enable = 0;

raw\_I2C\_data\_write(fd, data);

delayMicroseconds(10);

data.field.enable = 1;

raw\_I2C\_data\_write(fd, data);

delayMicroseconds(10);

data.field.enable = 0;

raw\_I2C\_data\_write(fd, data);

delayMicroseconds(100);

}

void I2C\_send\_value(const u\_int8\_t fd, u\_int8\_t value, u\_int8\_t ctrl\_option) {

I2C8bitData data = {.raw\_data = 0};

// send high

data.raw\_data = (value & 0xf0) | ctrl\_option;

I2C\_data\_write(fd, data);

// send low

data.raw\_data = ((value & 0x0f) << 4) | ctrl\_option;

I2C\_data\_write(fd, data);

}

void initialize\_lcd(u\_int32\_t fd) {

I2C8bitData data = {.raw\_data = 0};

I2C\_data\_write(fd, data);

data.field.bus4 = 1;

data.field.bus5 = 1;

I2C\_data\_write(fd, data);

I2C\_data\_write(fd, data);

I2C\_data\_write(fd, data);

data.field.bus4 = 0;

I2C\_data\_write(fd, data);

// Transmission mode change

I2C\_data\_write(fd, data);

data.field.bus7 = 1;

data.field.bus5 = 0;

I2C\_data\_write(fd, data);

show\_i2c\_data(data);

/\*

// Display off

data.raw\_data = 0;

I2C\_data\_write(fd, data);

data.field.bus7 = 1;

I2C\_data\_write(fd, data);

// Display clear

data.raw\_data = 0;

I2C\_data\_write(fd, data);

data.field.bus4 = 1;

I2C\_data\_write(fd, data);

\*/

// Entry mode

data.raw\_data = 0;

I2C\_data\_write(fd, data);

data.field.bus6 = 1;

data.field.bus5 = 1;

I2C\_data\_write(fd, data);

show\_i2c\_data(data);

}

int main() {

u\_int32\_t fd = wiringPiI2CSetup(0x27);

printf("[INFO] Start I2C Connection. fd = %d\n", fd);

initialize\_lcd(fd);

I2C\_send\_value(fd, 'A', 1);

}