int main (void) {

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_XTAL\_16MHZ | SYSCTL\_OSC\_MAIN);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOB); //B

ROM\_GPIOPinTypeGPIOOutput(GPIO\_PORTB\_BASE, GPIO\_PIN\_3); //pinMode(\_rst, OUTPUT);

ROM\_GPIOPinTypeGPIOOutput(GPIO\_PORTB\_BASE, GPIO\_PIN\_6);

ROM\_GPIOPinTypeGPIOInput(GPIO\_PORTB\_BASE, GPIO\_PIN\_7 ); //Configure pin as input

//Configure chosen pin for interrupts

ROM\_GPIOIntTypeSet(GPIO\_PORTB\_BASE, GPIO\_PIN\_7, GPIO\_BOTH\_EDGES); //Interrupt triggered on both edges (rising/falling distinguished in interrupt handler)

//Enable interrupts (on pin, port, and master)

GPIOIntEnable(GPIO\_PORTB\_BASE, GPIO\_PIN\_7);

ROM\_IntEnable(INT\_GPIOB);

ROM\_IntMasterEnable();

//Configure UART interrupts

//ROM\_IntEnable(INT\_UART0);

//ROM\_UARTIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT);

//ROM\_IntMasterEnable(); //Already done above for IR interrupts

ConfigureUART0();

UARTprintf("Turning on now\n");

ConfigureUART1();

ConfigureSSI();

ROM\_IntEnable(INT\_UART1);

ROM\_UARTIntEnable(UART1\_BASE, UART\_INT\_RX | UART\_INT\_RT);

begin();

UARTprintf("Turning on now2\n");

//setup();

//Setup OLED

lcdTestPattern();

ROM\_SysCtlDelay(SysCtlClockGet()/6); //delay(500);

initHW();

setTextSize(1);

fillScreen(BLACK);

drawLine(0, 64, 128, 64, GREEN);

//TODO Draw horizontal line across center

//Set up SysTick

ROM\_SysTickPeriodSet(RELOAD\_VALUE); //When SysTick is written to, this is written in as the reload value

ROM\_SysTickIntDisable(); //Default SysTick interrupt just reloads SysTick

ROM\_SysTickEnable(); //SysTick will always be running, it gets cleared at the start of the pulse

//Pulse variables

int signalArr[90];

int signalI=0;

float gap = 0.0;

float sigNum1 = 0.0;

int sigNum = 0;

int timesPressed = 0;

ROM\_SysTickIntDisable();

//int repeat = 0;

//TEXTING variables

unsigned char timeoutBegin = false; //boolean

char prevPress;

char\* currSet;

while(1)

{

ROM\_SysCtlSleep();

//UARTprintf("started: %c, ValueGet: %d", started, ROM\_SysTickValueGet());

if ((started == true) && (ROM\_SysTickValueGet() < (RELOAD\_VALUE - 2000000)))

{

//UARTprintf("I got in here yo");

GPIOIntDisable(GPIO\_PORTB\_BASE, GPIO\_PIN\_7);

timesPressed++;

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

gap = edgeTimes[i] - edgeTimes[i+1];

sigNum1 = gap/22550;

float sigNum2 = roundf(sigNum1);

sigNum = sigNum2;

for (int j=0; j < sigNum; j++){

signalArr[signalI+j] = (i%2==0)? 0 : 1; //if it's edgeTime[even] - edgeTime[odd], append low values, and vice versa

}

signalI += sigNum;

}

signalI--;

int binIndex = 0;

int signalIndexTwo = 0;

int dValue = 0;

int binArray[50];

for(int i = 0; i < 50; i++)

binArray[i] = -2;

while(signalIndexTwo < 84)

{

if((signalArr[signalIndexTwo] - signalArr[signalIndexTwo + 1]) == -1) //Rising edge, 0 to 128

{

binArray[binIndex] = 1;

signalIndexTwo = signalIndexTwo + 2;

binIndex++;

}

else if(signalArr[signalIndexTwo] - signalArr[signalIndexTwo + 1] == 1) //Falling edge, 128 to 0

{

binArray[binIndex] = 0;

signalIndexTwo = signalIndexTwo + 2;

binIndex++;

}

else

{

binArray[binIndex] = -1; //-1 if no change (no edge in period)

signalIndexTwo = signalIndexTwo + 2;

binIndex++;

}

} //Above loop will fill binArray with the values 1, 0, and -1 based on rising, falling, or neither

binIndex = 0;

int signalB = 0;

for(int i = 0; i < 50; i++)

{

if(binArray[i] == -2)

i = 50;

else

signalB++; //Holds # of elements, not address at last element

}

dValue = (8 \* binArray[signalB - 4]) + (4 \* binArray[signalB - 3]) + (2 \* binArray[signalB - 2]) + (binArray[signalB - 1]);

UARTprintf("%d ", dValue);

//DVALUE HAS INTEGER NUMBER OF WHAT THEY PRESSED

if ((dValue >= 2 && dValue <= 9) || dValue == 0) { //If dValue is one of the acceptable numkeys (otherwise do nothing)

//Record prevPress before updating currPress

prevPress = currPress;

currPress = (char)(((int)'0')+dValue);

//UARTprintf("CURRPRESS: %c\n", currPress);

//Select the correct char array for currSet

switch (currPress) {

case '2':

//strcpy(currSet, set2); // = set2;

currSet = "abc";

break;

case '3':

//strcpy(currSet, set3);

currSet = "def";

break;

case '4':

//strcpy(currSet, set4);

currSet = "ghi";

break;

case '5':

//strcpy(currSet, set5);

currSet = "jkl";

break;

case '6':

//strcpy(currSet, set6);

currSet = "mno";

break;

case '7':

//strcpy(currSet, set7);

currSet = "pqrs";

break;

case '8':

//strcpy(currSet, set8);

currSet = "tuv";

break;

case '9':

//strcpy(currSet, set9);

currSet = "wxyz";

break;

case '0':

//strcpy(currSet, set0);

currSet = " ";

break;

}

UARTprintf("CurrSet: %s", currSet);

//char\* death = "Death Does not Scare me";

//UARTprintf("PREVPRESS: %c\n", prevPress);

//testdrawtext(death, WHITE);

if (prevPress == '-') { //First keypress

setInd = 0;

currChar = currSet[setInd];

setCursor(drawX, drawY);

writeChar1(currChar, BLACK, WHITE);

setCursor(drawX, drawY);

//UARTprintf("Hanging?");

UARTprintf("%c", currChar);

//timeoutBegin = true;

}

else if (currPress == prevPress) { //If the same key was pressed again, cycle to next char in the set

setInd++;

//Wraparound setInd depending on the current set

if ((currPress == '2' || currPress == '3' || currPress == '4' || currPress == '5' || currPress == '6' || currPress == '8') && setInd > 2)

setInd = 0;

if ((currPress == '7' || currPress == '9') && setInd > 3)

setInd = 0;

if (currPress == 0 && setInd > 0)

setInd = 0;

//Update currChar and update it to the screen in WB

currChar = currSet[setInd];

setCursor(drawX, drawY);

writeChar1(currChar, BLACK, WHITE);

setCursor(drawX, drawY);

UARTprintf("%c", currChar);

//Set timeoutBegin flag to begin timing for timeout

//timeoutBegin = true;

}

else { //Otherwise, a different key was pressed, so set currChar in BW, and move the draw cursor

setCursor(drawX, drawY);

writeChar1(currChar, WHITE, BLACK);

ROM\_UARTCharPut (UART1\_BASE, currChar);

//SEND CHAR TO TX

UARTprintf("%c", currChar);

updateDraw();

//Then draw newest keypress

setCursor(drawX, drawY);

setInd = 0;

currChar = currSet[setInd];

setCursor(drawX, drawY);

writeChar1(currChar,BLACK, WHITE);

setCursor(drawX, drawY);

UARTprintf("%c", currChar);

}

}

//End texting

ROM\_SysCtlDelay (ROM\_SysCtlClockGet()\*.14);

GPIOIntEnable(GPIO\_PORTB\_BASE, GPIO\_PIN\_7);

//Reset global variables (edgeTimes does not need to be reset, gets overwritten)

edgeI = 0;

started = false;

//Reset other values

signalI =0;

for(int i = 0; i < 90; i++)

signalArr[i] = 0;

for(int i = 0; i < 200; i++)

edgeTimes[i] = 0;

}

}

}