Proximity Card Time Logger

Ignacio Díaz-Guardamino García (id5)

May 8, 2018
Chapter 1

Abstract

This project consisted on the creation of a compact system that could keep track of the time students spend in the lab. This is done by having them swipe their student cards upon entry and exit. The system will display on the screen the time spent on the current period, and at the end of each period will print through the thermal printer each student’s time in the lab. The system is also capable of opening the lab door when connected to a electrically operated door. In order to read proximity cards, the system will make use of a HID card reader.

To communicate with the peripherals, different communication protocols have been used: the thermal printer receives data via RS-232 serial communication, the Screen uses standard SPI and the HID card reader uses a proprietary two wire communication system.
Chapter 2

How to use the device

2.1 Students

For Students, the system is really simple. They just have to swipe their cards when entering or leaving lab, the system will display the time the student has spent in the lab in the current period.

2.2 Professors and TAs

For administrator use, the system is also easy to use, however, it requires some setup:
To set an administrator card, swipe it while holding the button in the back. After this, when the card is swiped everyone’s timer will be printed and set back to 0 ready for the next period.

The system has also the ability to open doors equipped with standard electric strike. To grant door access: when switch 1 in the back is closed, all the cards swiped will be granted door access. After this, the switch should be set back to open.

In case of power failures, the system can log to paper the times tracked. To enable automatic printing switch 2 should be set to it’s open position.

The system has been enabled to use ESP8266 wifi modules. To use them, the printer should be disconnected and switch 3 set to its closed position.
Chapter 3

State of the Art

What has currently been done to keep track of students’ times is having them write the time they enter and leave the lab in a table with the days of the week:

However, this system allows for students to cheat by writing down Times where they were not in the lab.

Similar devices to the one being developed already exist in the market, for example: the SEIKO TP-50. It makes use of cardboard cards where it prints the time spent in the lab. However, this will require giving the students cards and collecting them every week, which is not the best solution.
More advanced machines such as timeQplus make use of some sort of intelligent cards such as magnetic stripe, chip or proximity. However, this machines would require giving extra cards to the students. This is the main reason for building this system, using the university cards students already have to log their lab hours.
Chapter 4

Hardware

4.1 Block Diagram

4.2 Components

4.2.1 Microcontroller

The microcontroller used in this project is the ARM LPC1114FN28, the one provided in ECE 395 that suits this project because of the amount of IO pins available and the existence of timers with interruptions as well as serial communication to interface with the printer.

4.2.2 Hid Sensor

The proxpoint plus reader was used for this project, it communicates through a proprietary two way interface in which, when a card is scanned, it sends short pulses on both wires, the wire the pulses are sent indicates if that binary digit is a 1 or a 0, when a card is scanned, 35 bits are sent that contain the card number in the back. However, we have to mask it and divide it by two in order to get the number in the back of the card.
4.2.3 Thermal printer

The thermal printer uses a special kind of paper that when heated turns black. This, done in a controlled way lets us print characters.

A RS232 serial interface is used to send the data to the printer, and the chip built into it turns the ASCII characters sent into readable text.

![Thermal printer image]

4.2.4 Screen

The screen is a DOGXL240-7 LCD that works as Ram memory, it is connected via SPI, and to change pixels, we have to send, first the address that corresponds to the area of the screen we want to change followed by the data to ”store” in memory that correspond to the pixel values in that area. a code from the previous semester is used to generate the characters needed to display text in the screen.

4.2.5 Relay

As the system was already identifying users, an add-on for the project was to create a relay system with terminals to interact with a electric door strike. The system is switched off by default but the set up can be done with the flip of a switch.

![Relay image]
4.2.6 PCB

A schematic was designed using the software eagle to join all the components in one circuit. Then it was turned into a PCB also using eagle, it was manufactured on pcbway. It holds all the internal components and connectors for the external ones. One feature was to install a prototyping area in the middle to later on add more hardware in case it is needed.
Chapter 5

Next Steps

5.1 Enclosure

Even though the project is working, to make it usable in the lab, a enclosure could be built to make it look better.

5.2 Wireless connectivity

The project was originally designed with the objective of being totally detached from the Internet and only relying on the printer when displaying time information, however, when designing the PCB seemed like a good idea to add the connections for an ESP8266 WiFi module, and the C program is already built to interface with this module, this can be toggled by means of switch 3 as explained in the instructions chapter. However, code for the ESP8266 should be developed to interact with a web server or for example, make a service like IFTTT store times in a Google docs spreadsheet.

5.3 SD card

In case wireless is not available due to campus restrictions, there is a unpopulated SD card socket in the back of the PCB that, after adding an SD card could be used to store the data instead of having to use paper.
Chapter 6
Appendix

6.1 Main

```c
#include <stdio.h>
#include <rt_m misc . h>
#include "LPC11xx . h"

#include "font16x8 . h"

#define CLK_CONTROL_W 0x8f
#define DISP_RW 4

#define CARDM 100

typedef struct{
    int cnum;
    int time;
    int in;
    int intime;
    int door;
} T_CARD;
T_CARD allcards[CARDM];

extern void SER_init ( void );
extern int getkey ( void );
int cardNum = 0;
int bitNum = 0;
char name[32];
int rss = 0;
int c=0;
int z3=0;
int z1=0;
int sz=0;
int i;

/∗
* configureGPIO
* DESCRIPTION: sets up the GPIO pins
*/

void configureGPIO (){
    // enable clocks to GPIO block
    LPC_SYSCON->SYSAHBCLKCTRL |= (1UL << 6);
    LPC_SYSCON->SYSAHBCLKCTRL |= (1UL << 16);

    // set GPIO ports 0_3 and 0_1 to 0 (input) (data from card reader)
    LPC_GPIO0->DIR &= ~(1<<1);
    LPC_GPIO0->DIR &= ~(1<<3);

    // set gpio1−1 and 1−2 ans imput for the swiches
    LPC_GPIO1->DIR &= ~(1<<1);
    // give door access
    LPC_GPIO1->DIR &= ~(1<<2);
    // set continuos print.
    LPC_GPIO0->DIR &= ~(1<<5);
    // set system admin
    LPC_GPIO0->DIR &= ~(1<<11);
    // always print

    // buzzer
    LPC_GPIO0->DIR |= (1<<4);
    LPC_GPIO0->DATA |= (1<<4);

    // green led + door
    LPC_GPIO0->DIR |= (1<<7);
    LPC_GPIO0->DATA &= ~(1<<7);

}

/*/  
* timerSetup
* DESCRIPTION: sets up interrupts for the timer
*/  

void timerSetup (){  
    LPC_SYSCON->SYSAHBCLKCTRL |= (1<<7);  

LPC_TMR16B0->PR = 47;  // prescaling
LPC_TMR16B0->MCR = (3<<3);
LPC_TMR16B0->CCR = 0;
LPC_TMR16B0->MR1 = 10-1;
NVIC_EnableIRQ(TIMER_16_0_IRQn);
// interruption function
LPC_TMR16B0->TCR = 1;

/*
 * configureSPI
 * DESCRIPTION: sets up SPI
 */
void configureSPI()
{
  // Toggle SPI reset
  LPC_SYSCON->PRESETCTRL |= (1UL);

  // Enable clock for SPI0
  LPC_SYSCON->SYSAHBCLKCTRL |= (1UL << 11);

  // Enable SPI0_PCLK
  LPC_SYSCON->SSPOCLKDIV |= (1UL);

  // Enable clock for IOCONFIG
  LPC_SYSCON->SYSAHBCLKCTRL |= (1UL << 16);

  // Set data size to 9 bits
  LPC_SSP0->CR0 |= (1UL << 3);

  // Set clock high between frames
  LPC_SSP0->CR0 |= (1UL << 6);

  // Captures serial data on the transition back to the inter-frame
  LPC_SSP0->CR0 |= (1UL << 7);

  // Set location for CLK0
  LPC_IOCON->SCK_LOC |= (1UL << 1);  // SCK0 LOC

  // Set function for SPI0 pins
  LPC_IOCON->PIO0_2 |= (1UL);  // SSEL0
  LPC_IOCON->PIO0_6 |= (1UL << 1);  // SCK0
  LPC_IOCON->PIO0_9 |= (1UL);  // MOSI0

  // Set CPSDVSR to 8

LPC_SSP0→CPSR |= (1UL << 3);

// Enables the SPI
LPC_SSP0→CR1 |= (1UL << 1);
// NOTHING HERE

/**
 * writeSPI
 * DESCRIPTION: sends 9-bits of data through SPI
 * INPUTS: data—data to be written
 * bits—8 bits or 9 bits
 * RETURN VALUE: none
 * SIDE EFFECTS: fills the SPI data register
 */
void writeSPI(uint16_t data, int bits)
{
    while ((((1UL << 1) & LPC_SSP0→SR) == 0){}
LPC_SSP0→DR |= (data & 0x1FF);

    // DATA WRITE
    // 1. check the TNF bit in the status register (if FIFO is full)
    //——LPC_SSP0→SR (check bit 1)
    // 2. if FIFO is not full, write data into LPC_SSP0→DR (right justified)
}

/**
 * readSPI
 * DESCRIPTION: untested function
 * INPUTS: none
 * RETURN VALUE: 8 bits of data
 * SIDE EFFECTS: none
 */
char readSPI()
{
    char data;
    // wait for RNE to be high
    while (((1UL << 2) & LPC_SSP0→SR) == 0){}
data = LPC_SSP0→DR |= (data & 0xFF);

    return data;
    // DATA READ
    // 1. wait until the RNE bit in the status register is 1(if FIFO is not empty)
    //——LPC_SSP0→SR (check bit 2)
    // 2. if FIFO is full, read data from LPC_SSP0→DR (right justified)
}
void setupLCD ()
{
    /* Set the reset pin low */
    // LPC_GPIO0→DATA &= ~(1<<4);
    // Wait >1ms
    waitSysTick(0x0000D000);
    // Set reset pin high
    // LPC_GPIO0→DATA |= (1UL << 4);
    // Wait >150ms
    waitSysTick(0x00728000);*/
    // Set pump control
    writeSPI(0x002F, 9);
    // Set com end
    writeSPI(0x00F1, 9);
    writeSPI(159, 9);
    // Set display line start
    writeSPI(0x00F2, 9);
    writeSPI(0, 9);
    // Set display line end
    writeSPI(0x00F3, 9);
    writeSPI(159, 9);
    // Set line rate
    writeSPI(0x00A2, 9);
    // Set PM (contrast)
    writeSPI(0x0081, 9);  // Command INIT
    writeSPI(0x00B4, 9);  // Set PM to 128
    // Set display to 1 bit for 1 pixel
    writeSPI(0x00D1, 9);
    // Set auto increment
    writeSPI(0x0089, 9);
    // Set LCD mapping (MY mode)
    writeSPI(0x00C0, 9);
    writeSPI(0x0004, 9);
    // Set col address
    writeSPI(0x0000, 9);
    writeSPI(0x0010, 9);
    // Set page address
    writeSPI(0x0060, 9);
    writeSPI(0x0070, 9);

    // Enable display
    writeSPI(0x00A9, 9);
}

/*
 * writeChar
 * DESCRIPTION: writes one character to the screen
 * INPUTS: letter—character to write
 */
void writeChar(char letter, char page, char col){
    int offset, i;
    uint16_t byte;
    // Convert ASCII letter to integer offset
    offset = letter;

    // Set col address
    writeSPI(col & 0x000F, 9);
    writeSPI(0x0010 | ((col & 0x00F0) >> 4), 9);
    // Set page address
    writeSPI(0x0060 | (page & 0x000F), 9);
    writeSPI(0x0070 | ((page & 0x00F0) >> 4), 9);

    // Write top row of character
    for(i = 0; i < 16; i += 2){
        byte = font[16*offset + i];
        writeSPI(0x0100 | (byte & 0x00FF), 9);
    }

    // Increment page and reset col address
    page++;
    writeSPI(col & 0x000F, 9);
    writeSPI(0x0010 | ((col & 0x00F0) >> 4), 9);
    writeSPI(0x0060 | (page & 0x000F), 9);
    writeSPI(0x0070 | ((page & 0x00F0) >> 4), 9);

    // Write bottom row of character
    for(i = 1; i < 16; i += 2){
        byte = font[16*offset + i];
        writeSPI(0x0100 | (byte & 0x00FF), 9);
    }
}

/*
 * writeString
 * DESCRIPTION: writes full string to screen
 * INPUTS: string – pointer to string
 *    page – row to write on
 *    col  – column to write on
 * RETURN VALUE: none
 * SIDE EFFECTS: none
 */
void writeString(char* string, char page, char col){
# clearScreen

*DESCRIPTION*: wipes screen

*INPUTS*: none

*RETURN VALUE*: none

*SIDE EFFECTS*: writes over screen RAM

```c
void clearScreen()
{
    int i;
    writeSPI(0x00A8, 9);
    // Set col address
    writeSPI(0x0000, 9);
    writeSPI(0x0010, 9);
    // Set page address
    writeSPI(0x0064, 9);
    writeSPI(0x0070, 9);
    for (i = 0; i < 240*16; i++) {
        writeSPI(0x0100, 9);
    }
    //makeStamp();
    writeSPI(0x00A9, 9);
}
```

# displayInt

*DESCRIPTION*: displays card number on the screen

*INPUTS*: in – card number

*RETURN VALUE*: none

*SIDE EFFECTS*: none

```c
void displayInt(int in){
    int i;
    char buf[7];
```
// convert from int to ASCII character
for (i = 5; i >= 0; i--){
    buf[i] = in%10 + 48;
    in /= 10;
}
if (buf[0]==48){
    buf[0]='.';
}
buf[6] = '\0';
// writeString("Your card number is: ", (char)11, (char)40);
writeString(buf, (char)11, (char)90);

void displaytime(int time){
    int hours=time/60;
    int minutes=time%60;
    if (hours>9){
        writeChar(((hours/10)%10)+48),(char) 15, (char) 168);
        writeChar(((hours%10)+48),(char) 15, (char) 176);
    }else{
        writeChar(((hours%10)+48),(char) 15, (char) 176);
    }
    if (minutes>9){
        writeChar(((minutes/10)%10)+48),(char) 17, (char) 168);
        writeChar(((minutes%10)+48),(char) 17, (char) 176);
    }else{
        writeChar(((minutes%10)+48),(char) 17, (char) 176);
    }
}
int findcard(int cardNum){
    int index=0;
    for (i=1;i<CARDM; i++){
        if(allcards[i].cnum == cardNum){
            index=i;
            break;
        }
    }
    if(index==0){
        for (i=1;i<CARDM; i++){
            if(allcards[i].cnum == 0){
                index=i;
            }
        }
    }
}
allcards[i].cnum = cardNum;
allcards[i].time = 0;
allcards[i].door = 0;
allcards[i].in = 0;
break;
}
}
if(index==0){
    for(i=1;i<CARDM;i++){
        if(allcards[i].door == 0){
            if(allcards[i].time == 0){
                if(allcards[i].in == 0){
                    index=i;
                    allcards[i].cnum = cardNum;
                    break;
                }
            }
        }
    }
}
}
return index;
}

void opendor(void){
    LPC_GPIO0->DATA |= (1<<7);
    for (i = 0; i < 0x0004FFFF; i++)
    {
        LPC_GPIO0->DATA &= ~(1<<7);
    }
}

void printtimes(void){
    int i;
    printf("\n\rCard_Times.\n\r\n\rClass: \n\rWeek: \n\r" );
    for(i=1;i<CARDM;i++){
        if(allcards[i].cnum != 0){
            if(allcards[i].time != 0){
                printf("%05d-%02d:%02d#\n", allcards[i].cnum , allcards[i].time , allcards[i].in );
            }
        }
    }
    printf("\n\r\n\r");
}

void cardhandler(int cardNum){
    int index;
    clearScreen();
    clearScreen();
// take only bits relevant from the cardID
cardNum = cardNum >> 1;
cardNum &= 0x0FFFF;
writeString("Your cardnumber is ", (char) 9, (char) 40);
displayInt(cardNum);
if (!(LPC_GPIO0->DATA & (1<<11))){
    printf("%06d\n", cardNum);
}
index = findcard(cardNum);
if(index==0){
    writeString("Memory full.", (char) 15, (char) 8);
} else {
    if(LPC_GPIO0->DATA & (1<<5)){
        allcards[index].door=2;
    } else if(allcards[index].door > 0){
        opendor();
    } else if(!(LPC_GPIO1->DATA & (1<<1))){
        allcards[index].door=1;
    }
    if(allcards[index].door == 2){
        printtimes();
    } else {
        if(allcards[index].time%60==1){
            writeString("minute_", (char) 17, (char) 184);
        } else {
            writeString("minutes_", (char) 17, (char) 184);
        }
    } if(allcards[index].time/60==1){
        writeString("hour_", (char) 15, (char) 184);
    } else {
        writeString("hours_", (char) 15, (char) 184);
    }
    if(allcards[index].in==0){
        allcards[index].in=1;
        allcards[index].intime=0;
        writeString("Welcome. Time:", (char) 15, (char) 8);
    } else {
        allcards[index].in=0;
        allcards[index].time += allcards[index].intime;
        writeString("Leaving Lab. Time:", (char) 15, (char) 8);
    }
    if(LPC_GPIO1->DATA & (1<<2)){
        printf("%d−actual:%02d total:%02d\n", allcards[index].cnum, allcards[index].intime /60, allcards[index].intime %60, allcards[index].time /60, allcards[index].time %60);
    }
allcards[index].intime=0;
}
displaytime(allcards[index].time);
}

rss=10;
}

void updatetime(void){
   for (i=0; i<CARDM; i++){
      if (allcards[i].in != 0){
         allcards[i].intime++;
         if (allcards[i].intime>500){
            allcards[i].in = 0;
            allcards[i].intime = 0;
         }
      }
   }
}
/*
 * PIOUS0_IRQHandler
 * DESCRIPTION: interrupt handler
 * INPUTS: none
 * RETURN VALUE: none
 * SIDE EFFECTS: none
*/

void TIMER16_0_IRQHandler(void)
{
   if (LPC_TMR16B0->IR & (1<<1)) //match1 interrupt?
   {
      LPC_TMR16B0->IR = (1<<1); //reset flag
      c++; //inc counter
      if (c%100000==0){
         if (rss>1){
            rss--;
         }
         if (sz>1){
            sz--;
         }
      }
      if (c>=6000000) //every minute
      {
         c=0;
         updatetime();
      }
}
```c
int main()
{
    timerSetup();
    SER_init();
    configureGPIO();
    configureSPI();
    setupLCD();
    clearScreen();
    printf("Starting.\n\r");
    writeString("Scan_Proxcard.", (char)11, (char)68);
    for (i=0;i<CARDM; i++){
        allcards[i].cnum =0;
        allcards[i].time =0;
        allcards[i].in =0;
        allcards[i].door =0;
    }
    while(1){
        if ((LPC_GPIO0->DATA & (1<1))){
            z1=0;
            if ((LPC_GPIO0->DATA & (1<3))){
                z3=0;
            }else{
                if (z3==0){
                    bitNum++;
                    cardNum = cardNum << 1;
                    cardNum++;
                    // printf("1");
                    z3=1;
                    sz=4;
                }
            }
        }else{
            if (z1==0){
                //} else{
                bitNum++;
                cardNum = cardNum << 1;
                // printf("0");
                z1=1;
                sz=4;
                // if Data1 line is high add a 1
            }
        }
        if (bitNum==35){
            cardhandler(cardNum);
            cardNum = 0;
        }
    }
}
```
bitNum = 0;
}
if (rss==1){
    rss=0;
    clearScreen();
    clearScreen();
    WriteString("Scan Proxcard.", (char)11, (char)68);
}
if (sz==1){
    sz=0;
    cardNum = 0;
    bitNum = 0;
}
if (! (LPC_GPIO1->DATA & (1<<1))&&!((LPC_GPIO1->DATA & (1<<1))){
    printf("\r Instructions:
\r *1: To set an admin swipe the card
\r *2: to grant door access: close switch 1, swipe the cards then
\r *3: to enable automatic printing: set switch 2 to its open position
\r *4: to enable wireless functionality: set switch 3 to its
\r")
}
}

6.2 Serial

#include "LPC11xx.h" /* LPC11xx definitions */

/*

   Initialize UART pins, Baudrate

*/

void SER_init (void) {

    /* configure PINs GPIO1.6, GPIO1.7 for UART */
    LPC_SYSCON->SYSAHBCLKCTRL |= ((1UL << 6) | /* enable clock for GPIO */
        (1UL << 16) ); /* enable clock for IOCON */

    /* P1.6 is RxD */
    LPC_IOCON->PIO1_6 = (1UL << 0);
    /* P1.7 is TxD */
    LPC_IOCON->PIO1_7 = (1UL << 0);

    /* configure UART0 */
    LPC_SYSCON->SYSAHBCLKCTRL |= (1UL << 12); /* Enable clock to UART */
    LPC_SYSCON->UARTCLKDIV = (4UL << 0); /* UART clock = CCLK / 4 */
LPC_UART->LCR = 0x83;  /* 8 bits, no Parity, 1 Stop bit */
LPC_UART->DLL = 24;      /* 115200 Baud Rate @ 12.0 MHZ PCLK */
LPC_UART->FDR = 0x85;    /* FR 1.627, DIVADDVAL 5, MULVAL 8 */
LPC_UART->DLM = 0;       /* High divisor latch = 0 */
LPC_UART->LCR = 0x03;    /* DLAB = 0 */
}

/*  Write character to Serial Port 

int sendchar (int c) {
    while (!(LPC_UART->LSR & 0x20));
    LPC_UART->THR = c;
    return (c);
}
*/

/*  Read character from Serial Port (blocking read)

int getkey (void) {
    while (!(LPC_UART->LSR & 0x01));
    return (LPC_UART->RBR);
}