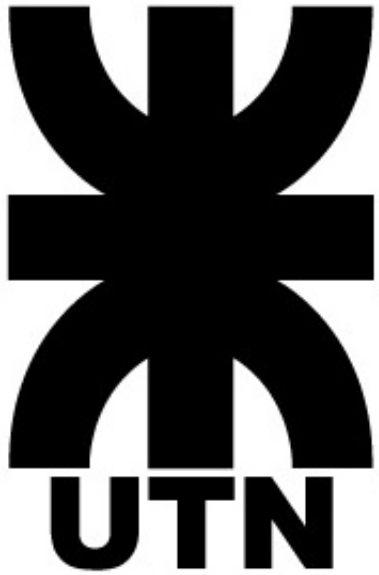

Universidad Tecnológica Nacional
Facultad Regional de Córdoba

Trabajo Práctico Final
Secuencia de leds sobre RaspberryPi

Navarro, Facundo 63809
Nobile, Jonathan Bleddyn 69325

Curso: 4R2
Grupo $N°13$

Técnicas Digitales II

Docentes:
Ing. Perez Paina, Gonzalo
Ing. Pereyra, Estefanía

25 de noviembre de 2019

Índice

1. Introducción

Este proyecto consiste en integrar dichos ejercicios de la siguiente manera:

- Realice un programa a fin de que el usuario pueda seleccionar desde un menú, una de ocho secuencias de luces posibles. Cuatro de ellas serán comunes para todos los proyectos y son: " El auto fantástico", "El choque", "La apiladaz "La carrera". Los otros cuatros serán propios de cada grupo y se deberán implementar dos de ellas con algoritmo y los dos restantes por medio de la técnica de tablas de datos.
- Implemente el control de acceso a este menú mediante password.
- Cada vez que el usuario seleccione una secuencia el programa deberá cambiar la pantalla para indicar cual secuencial está ejecutándose y cómo hacer para salir de la misma. Al optar por abandonar la actual, el programa deberá regresar al menú principal inmediatamente sin completar la secuencia que se está desarrollando y apagando todas las luces.
- Permita la posibilidad de controlar la velocidad de cada secuencia. Presionando la flecha hacia arriba se incrementará la velocidad y presionando la flecha hacia abajo se reducirá. Introduzca el censado de las teclas oprimidas en el lugar apropiado de su programa a fin de percibir la reacción del sistema en forma inmediata, independiente de la velocidad actual. La velocidad ajustada en cada secuencia deberá conservarse entre llamadas a diferentes secuencias.
- El valor inicial correspondiente a la velocidad de las secuencias deberá ingresarse mediante la lectura del estado de los potenciómetros que están conectados a las entradas analógicas del conversor A/D.
- Generar una opción en el programa que permita establecer dos modos de trabajo: local y remoto. En modo local las secuencias de luces se ejecutarán en los leds que se encuentran en el hardware adicionado a la placa Raspberry donde se ejecuta el programa. En modo remoto las secuencias se ejecutarán sobre el hardware adicional colocado en otra Raspberry y conectada a la que ejecuta el programa mediante un cable serie RS-232. Se podrá usar el mismo programa para implementar esta opción en las dos Raspberry o realizar uno principal y otro secundario.
- Como opción genere una sección destinada a establecer las velocidades iniciales de las secuencias realizando el ajuste de los potenciómetros.

2. Objetivos

Aplicar los conceptos aplicados durante el transcurso de la materia, aglutinar los puntos prácticos en un solo programa sobre la placa de desarrollo RaspberryPi.

3. Diagrama de flujo

EL programa al iniciar, pide una contraseña ya establecida, si la misma se ingresa erróneamente en 3 oportunidades el programa termina, caso contrario se pasa al menú principal.

En este nivel se visualiza un pequeño listado de las secuencias disponibles, valores de velocidad y *ADC*, como así también en un texto titilante se hace hincapié en las teclas para proseguir o finalizar el programa. Dentro del menu principal presionando:

- **Q** : Se finaliza el programa.
- **R**: Se conmuta a *modo remoto*, a través de la comunicación serie. Volviendo a presionar **R** se vuelve a *modo local*.
- **P** : Se pasa el valor leído del *ADC* (en porcentual) al valor de la variable "velocidad".
- **1 ... 8** : Se ingresa a la secuencia seleccionada.

Al pasar a alguna de las secuencias seleccionadas se puede modificar el valor de la velocidad apretando las teclas **"UP/DOWN"** , para finalizar la secuencia se debe presionar **Q** y se vuelve al menú principal.

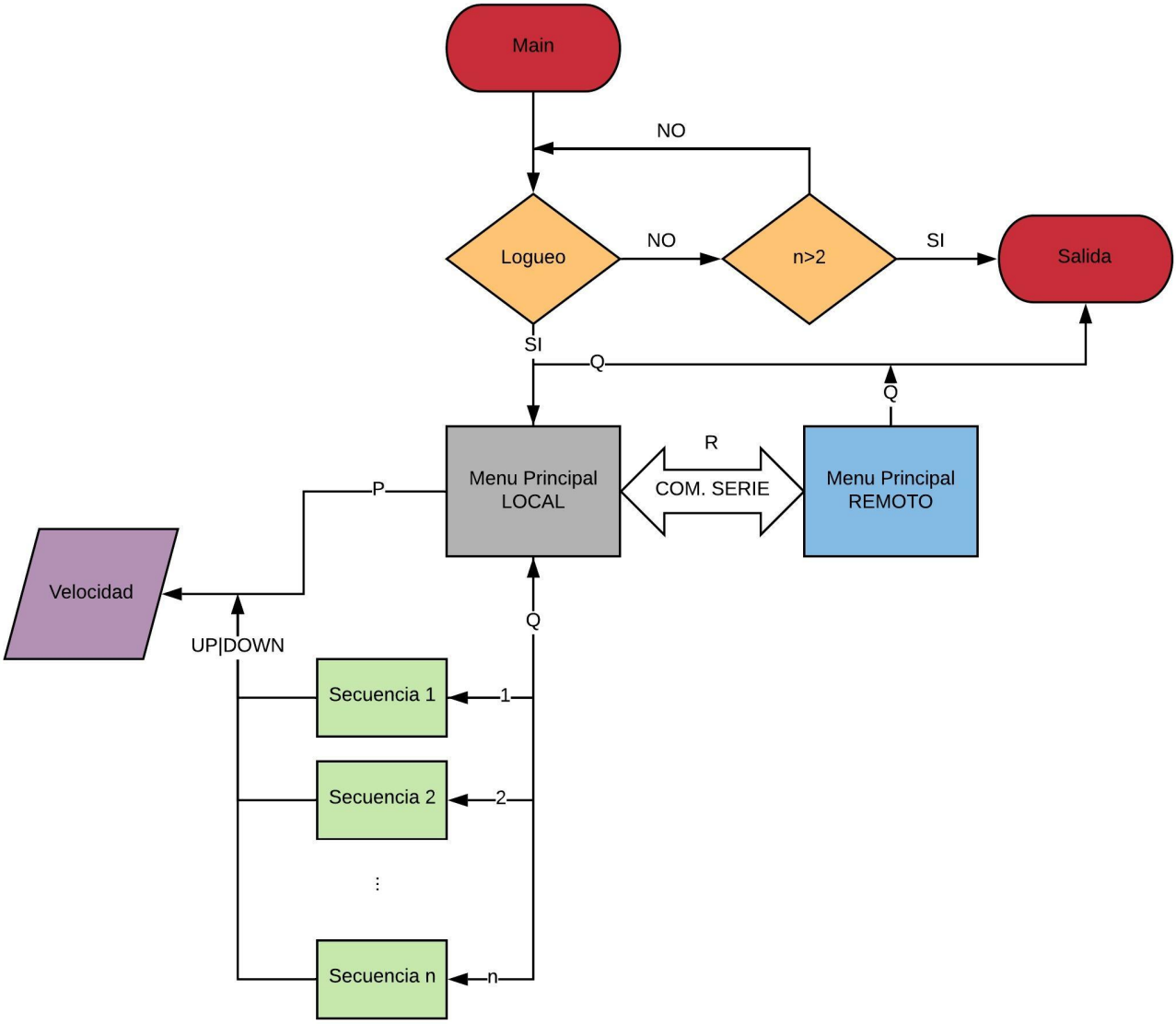

Figura 1: Diagrama de flujo del programa.

4. *main.c*

El archivo fuente *main.c* contiene la estructura principal del programa, a través del cual se llaman las funciones periféricas. A continuación se explicaran las partes fundamentales.

4.1. Inicialización de GPIO.

```
  led_config(leds);
```

Listing 1: Fragmento de main.c

```
#include <wiringPi.h>

void led_config(int *leds)
{
  wiringPiSetup();  // Inicializo las funciones de wiringPi, necesario segun documentacion
  pinMode(leds[0], OUTPUT);
  pinMode(leds[1], OUTPUT);
  pinMode(leds[2], OUTPUT);
  pinMode(leds[3], OUTPUT);
  pinMode(leds[4], OUTPUT);
  pinMode(leds[5], OUTPUT);
  pinMode(leds[6], OUTPUT);
  pinMode(leds[7], OUTPUT);
}
```

Listing 2: led_config.c

A penas se ingresa al programa se inicializa la configuración de los GPIO para que se comporten como salida, los mismos corresponden a los leds.

4.2. nCurses inicialización

```
  initscr();  // Inicializo mi pantalla de ncurses, por defecto llamada stdscr
  noecho();   // Deshabilito el echo automatico de los caracteres tipeados
  keypad(stdscr, TRUE);   // Deshabilito para que se pueda usar el keypad del teclado en
    la pantalla de ncurses
  initscr();    // Comienzo la interfaz de ncurses
  curs_set(0);    // Deshabilito el cursor de la pantalla.

  if(has_colors()){     // Si la terminal soporta colores
    start_color();      // Arranco ncurses en modo con colores
  }
```

Listing 3: Fragmento de main.c

4.3. Inicialización de ADC y captura de datos.

```
  adcCrudo(1, &ADC, &fotocelula, &termistor);

  potenciometro = modifier * (ADC+2);
  velocidad_ms  = 101 - 1 * potenciometro;
```

Listing 4: Fragmento de main.c

```
#include <stdlib.h>
#include <wiringPi.h> // -lwiringPi
#include <pcf8591.h> // -lbcm2835

#define Address 0x48 // i2cdetect -y 1 nos da este valor
#define BASE 64
#define A0 BASE+0
#define A1 BASE+1
#define A2 BASE+2
#define A3 BASE+3 // No se usa de momento, es para leer el valor externo
```

```
void adcCrudo(int inicializar, int *potenciometro, int *fotocelula, int *thermistor){
  if(inicializar) pcf8591Setup(BASE,Address);

  *potenciometro  = analogRead(A0);
  *fotocelula   = analogRead(A1);
  *thermistor   = analogRead(A2);
}
```

Listing 5: adcCrudo.c

El valor del potenciómetro se obtiene a través de la librerías pcf8591 y wiringPi, estas nos posibilidad con la función de *analogRead*(); de retornar el valor convertido en un rango de $8bit$, es decir el resultado sera de entre 0-255.

A este valor se lo multiplica por un factor de escalamiento "modifier", para linealizar el potenciómetro, para poder trabajar en modo porcentual $(0 - 100\,\%)$.

Como las funciones de las secuencias de leds se trabaja con microsegundos, y suponiendo un delay máximo de $100ms$ aproximadamente, se realiza la opreacion indicada en el Listing 5.

Estos valores (*potenciometro* y *velocidad_ms*) luego se pasaran a la pantalla.

4.4. Logueo.

```
  if(!logueo(intentos))
    return 0;
```

Listing 6: Fragmento de main.c

La función de logueo toma como parámetro los intentos permitidos, mediante las posibilidades que nos otorga nCurses se grafica una interfaz donde se pueda visualizar la entrada por teclado, y un resultado si el password es correcto o no, ya que logueo se definió como una función de tipo *booleano* , esta hará finalizar el programa en caso fallido o seguirá con su ejecución en caso de acierto.

Figura 2: Acceso al sistema.

4.5. Detección de tecla

Se escribió una función análoga a *kbhit*(), que detecta si se presiono alguna tecla y retorna el valor tecleado.

```
#include <ncurses.h>  // -lncurses

bool deteccionTecla(int caracter, bool ug){ // Funcion que detecta si el usuario toco
    escape, sin retrasar la ejecucion del programa
  int c;  // Caracter a recibir
  bool r; // Valor de retorno, que indica si recibi escape o no

  nodelay(stdscr, TRUE);  // Desactivo la espera a que se escriba un caracter,
        //   esto es lo que hace que no se pause la ejecucion
  c = getch();    // Veo si puedo recibir un caracter

  if(c!=ERR){   // Primero me fijo si recibi algun caracter
    if(c==caracter){  // Me fijo si ese caracter fue un escape
      r=1;
    } else{
      r=0;
      if(ug && c!=27){
        ungetch(c); // Lo devuelvo al buffer de caracteres de stdin
      }
    }
  }else{    // En caso de que nisiquiera haya recibido algun caracter
    r = 0;
  }

  nodelay(stdscr, FALSE); // Vuelvo a habilitar que la funcion getch espere a que el
    usuario teclee

  return(r);  // Devuelvo si recibi, o no, un escape
}
```

Listing 7: deteccionTecla.c

Uno de los aspectos fundamentales de esta parte del código es la función *no_delay*, la cual hace que *getch*() se transforme a una función no bloqueante, que devuelve un valor *ERR* , predefinido en *nCurses* si es que no se ha tipeado nada y devuelve el valor del carácter tipeado en caso contrario.

4.6. Modo local y remoto

Para pasar de un modo a otro basta con presionar la tecla **“R”**

```
    clear();
    mainmenu(potenciometro, speed, remoto);

    int puertochar;
    puertochar = rs232rx(fdpuerto);

    if(!remoto)
    {
      if (deteccionTecla('R',1))
        remoto = 1;
    }

    else
    {
      rs232escribo(fdpuerto);
      if (deteccionTecla('R',1))
        remoto = 0;
    }
```

Listing 8: Fragmento de main.c

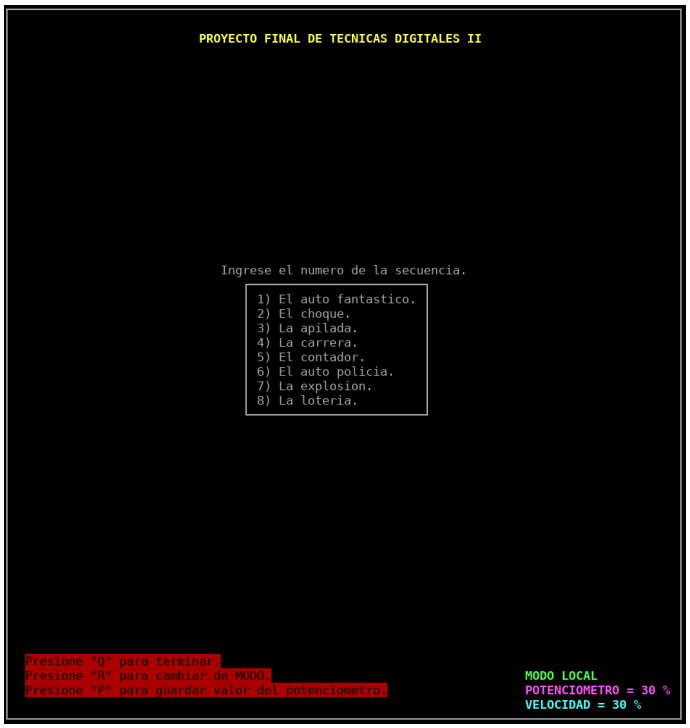

Figura 3: Modo local.

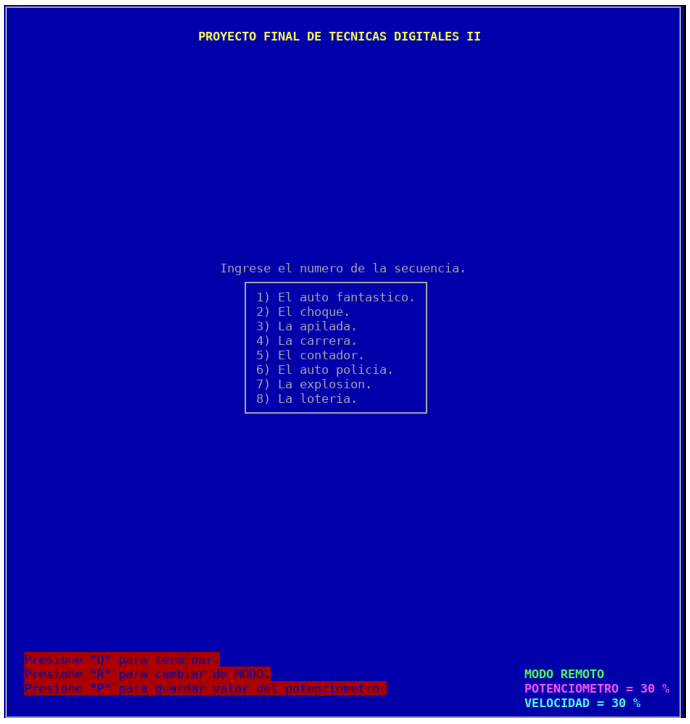

Figura 4: Modo remoto.

En cualquiera de los casos se modifica el valor de una variable bandera “remoto”, el cual determinar el color de la pantalla. Tanto los valores de remoto como potenciómetro y velocidad son pasados a la función *mainmenu*() la cual se encarga de graficar el menú principal.

En el modo remoto los valores teclados se pasaran a la otra Raspberry (también configurada en modo remoto) a través de una comunicación serial.

4.7. Secuencia

Una vez seleccionada la secuencia se pasa a una pantalla similar donde se observa la secuencia ejecutándose y la velocidad de la misma, esta puede ser modificada a través de las flechas del teclado.

```
#include <wiringPi.h> // -lwiringPi
#include <ncurses.h>  // -lncurses
#include "../headers/deteccionTecla.h"
#include "../headers/controlVelocidad.h"
#include "../headers/rs232.h"

int apilada(int leds[8],int periodo, int fdpuerto, bool remoto){
  bool valores[8] = {0,0,0,0,0,0,0,0}; // Cadena que guarda estados de los leds
  char programa[]  = "La apilada";
  int speed;

  for(int j=0; j<8; j++){ // Realizo dos ciclos, uno de izq. a derecha, otro de derecha a
    izq.
    for(int i=0; i<8-j+2; i++){ // Prendo todos los leds uno a la vez, en orden
      if(i<8-j){
        valores[i] = 1;
        if(i!=0) valores[i-1] = 0;
      }else{
        if(i==8-j) valores[i-1]=0;
        else valores[i-2]=1;
      }
      // Igualo todos los pines a sus respectivos valores
      if(!remoto){
        digitalWrite(leds[0], valores[0]);
        digitalWrite(leds[1], valores[1]);
        digitalWrite(leds[2], valores[2]);
        digitalWrite(leds[3], valores[3]);
        digitalWrite(leds[4], valores[4]);
        digitalWrite(leds[5], valores[5]);
        digitalWrite(leds[6], valores[6]);
        digitalWrite(leds[7], valores[7]);
      }
      for(int d=0; d<periodo; d++){
        int puertochar;
        if(remoto) rs232escribo(fdpuerto);
        puertochar = rs232rx(fdpuerto);
        speed = 101 - (0.9900 * periodo);
        periodo = controlVelocidad(periodo, puertochar);
        if(deteccionTecla('Q', 0) || puertochar=='Q'){
          ungetch('Q');
                j=8; i=8; break;
        }
        delay(1); // Delay entre ciclos
      }
        ledmenu(remoto, programa, speed);
    }
  }
  // Cuando termina el programa apagado todos los pines
  digitalWrite(leds[0], LOW);
  digitalWrite(leds[1], LOW);
  digitalWrite(leds[2], LOW);
  digitalWrite(leds[3], LOW);
  digitalWrite(leds[4], LOW);
  digitalWrite(leds[5], LOW);
  digitalWrite(leds[6], LOW);
  digitalWrite(leds[7], LOW);

  return periodo;
}
```

Listing 9: apilada.c

Luego del algoritmo para la ejecución correcta de las luces, se comprueba si el programa debe ser remoto o local,en caso que sea remoto local, se pasan estos parámetros a los pines de salida, por el otro lado si fuese remoto se recibe el valor de la tecla presionada desde la otra terminal.

Figura 5: Pantalla de una secuencia de luces.

5. Conclusión

Para lograr el desarrollo del trabajo integrador, se requirió de los conocimientos y usos de los puertos de entrada y salida de la Raspberry Pi 3, utilización de librerías para el integrado pcf859, y protocolo de comunicación, tanto $I2C$ como serial entre dos puertos.

A su vez crear encabezados y utilizar librerías que no habían sido necesarias para la primera presentación, con respecto al ordenador de placa reducida (Raspberry Pi 3), se observó y comprobó la versatilidad cuando se requiere velocidad y adaptación a otros terminales o periféricos.

En lo que respecta a conocimientos adquirido, se mejoró en la escritura de código, la fluidez a la hora de utilizar la terminal de software libre, los atajos, compilar, ejecutar y depurar con más eficiencia.