Notes on Home Treadmill

Manuel Escriche

May 22, 2021

Abstract

Some notes taken on how the treadmill at home was repaired by making a new controller.

This new controller was implemented by using an Esp32 SoC to control de motors and interact with human users. Esp32 SoC has been programmed in C language on top of ESP-IDF libraries stack which includes FreeRTOS.

These notes are intended to help when having to come back to fix any bug, repair any hardware or even improve it by adding new components.

Contents

1 Operative Know How

Setting the work environment

```
> alias
> get_idf
```

Configuring compilation

```
> idf.py menuconfig
```

It displays a screen where to select options that configure compiling directives used by components like Logs, RTOS, etc As an example the last version installed in the treadmill machine is compiled with No logs.

Compiling the program

```
> idf.py build
```

Cleaning workspace Sometimes weird things happen and there's need to clean up all content in the work space, then two commands are avaible:

```
> idf.py clean
> idf.py fullclean
```

Flashing into the mcu

```
> idf.py -p /dev/cu.SLAB_USBtoUART flash
> idf.py -p /dev/cu.SLAB_USBtoUART flash monitor
```

Getting help about esp-idf ESP-IDF Programming Guide

Getting help about RTOS FreeRTOS Documentation

2 How to modify speed programs

Speed programs are stored in the file `~/main/programs.c`, which need to be edited for any change. Look at the example below.

```
Program  test = {
  .owner= "MEV",
  .oseq = { {4,1,0}, {6,1,0}, {0,0,0} //end token
  }
};
```

It's important to assign a known owner, which are: `{"MEV", "MJGG", "MiEG", "MaEG"}` corresponding to `{"Manuel", "Marichu", "Miguel", "Manu"}`

The order sequence is composed of pieces holding three values `{speed, time, slope}`, whose units are subsequently `{Km/h, minutes, %inclination}`. The order sequence must finish with `{0,0,0}` in order to finish the program.

Optionally, you have to edit the file `~/main/session.c`, and specifically the sentences below in order to create new speed programs, or to remove old ones, but you don't need to touch it just for adjusting values.

```
extern Program test, mev1, mev2, mev3, mev4,
maeg1, maeg2, maeg3,  mieg1,  mjgg1, mjgg2, mjgg3;

static Program* book[] = {
  &maeg1, &maeg2, &maeg3, &mieg1,
  &mjgg1, &mjgg2, &mjgg3,
  &mev1, &mev2, &mev3, &mev4, &test,
};
```

In summary, the steps to take are:

1. Edit the file `programs.c`
2. Optionally edit the file `session.c`
3. Compile it: `> idf.py build`
4. and Flash it: `> idf.py -p /dev/cu.SLAB_USBtoUART flash`

3 Hardware

A PCB (=Printed Circuit Board) has been designed by using KiCad, have a look at the scheme in figure 1:

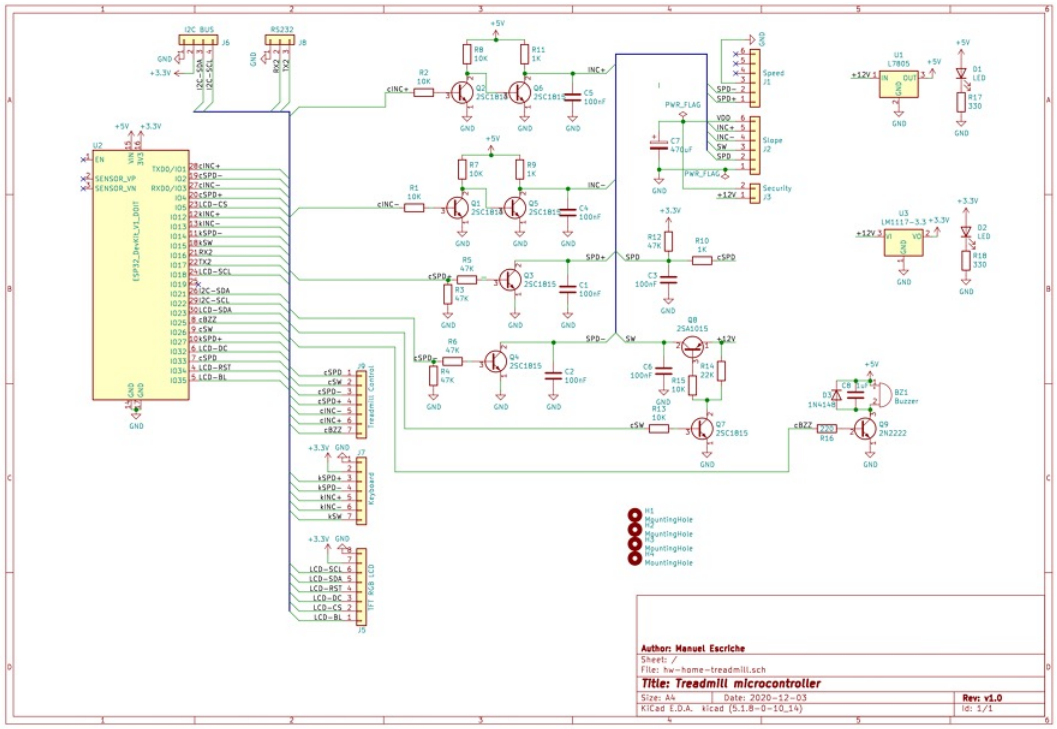

Figure 1: home treadmill schema

This PCB basically holds the interface to the power phase of the treadmill, and the mcu esp32, with connectors for a screen, a small keypad, an I2C bus and RS232 conection.

3.1 Treadmill Interface

Three connectors J1-Speed, J2-Slope and J3-Security conform the interface to the treadmill platform.

Power Input happens at the pins 6,1 (VDD, GND) of J2 .

Security connector J3 is connected to a micro switch spdt held on by a magnetic piece, which is taken off by the user in case of emergency. This micro switch cuts off pcb power through SW signal, which acts over VDD.

Speed input is taken by pulses on pin 2 of J2.

Speed control happens by using pins 1,2 of J1. These signals allow controlling speed: SPD+ speeds up engine, and SPD+ slows down it.

Slope control occurs by using pins 5,4 of J2. These signals allow setting slope: INC+ increases slope, and INC+ decreases it. Additionally J9-Treadmill control has been provided just for signal verification purposes, in order to verify electronics between mcu and treadmill.

3.2 Screen Interface

The connector J5-TFT RGB LCD is the physical interface to the screen. It holds pins for controlling an ST7735 128x160 screen. Its control is made effective through an SPI (Serial Peripheral Interface) using signals SCL(=clock), SDA(=MOSI) and CS(=chip select), leaving MISO out since the screen isn't producing any input.

Additionally there're the signals needed for the screen itself: RST(=reset), DC(=data/command), and BL(=backlight).

3.3 Keypad Interface

The connector J7-Keyboard conform the interface to the keypad. Initially it was thought to manage keys for speed, slope and stop/walk. However, it was modified to be connected to an I2C-GPIO Extension module since 8 keys were used in the keypad: Slope(2 keys), Speed(4 keys), Start/Stop(1 key), Pause(1 key).

3.4 RS232 and I2C Interfaces

IIC interface to the ambient sensor so that temperature and humidity can be measured and displayed on screen

RS232 interface to connect a Rasperry Pi3 meant to display speed programs and speed evolution.

3.5 PCB Implementation

KiCaD PCB has been used to design the PCB. Its manufacturing was done by PCBWay

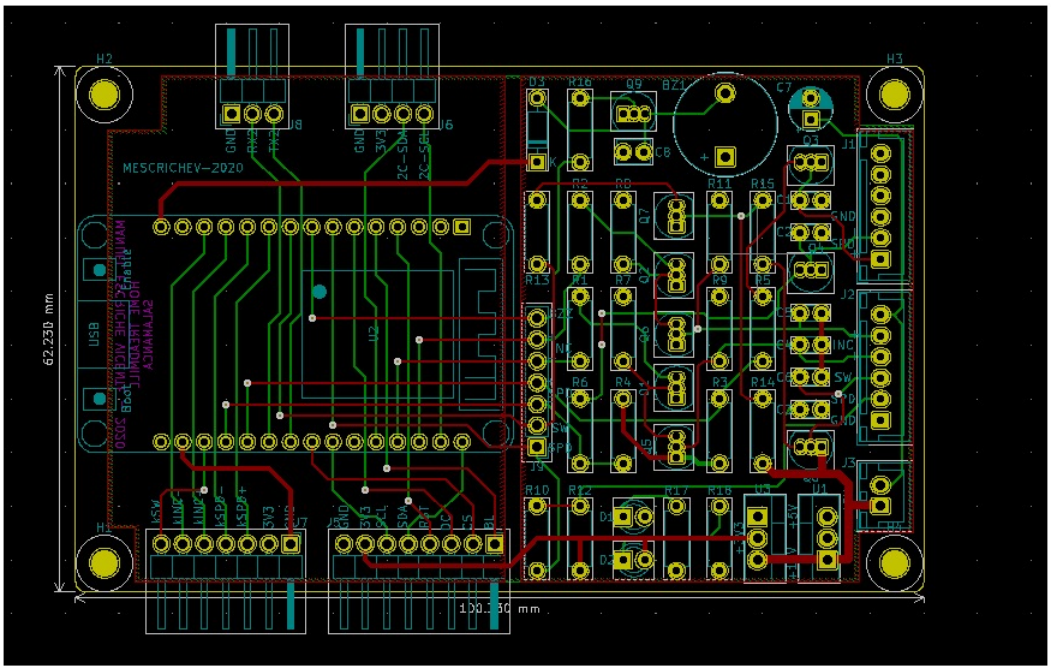

Figure 2: home treadmill pcb - design

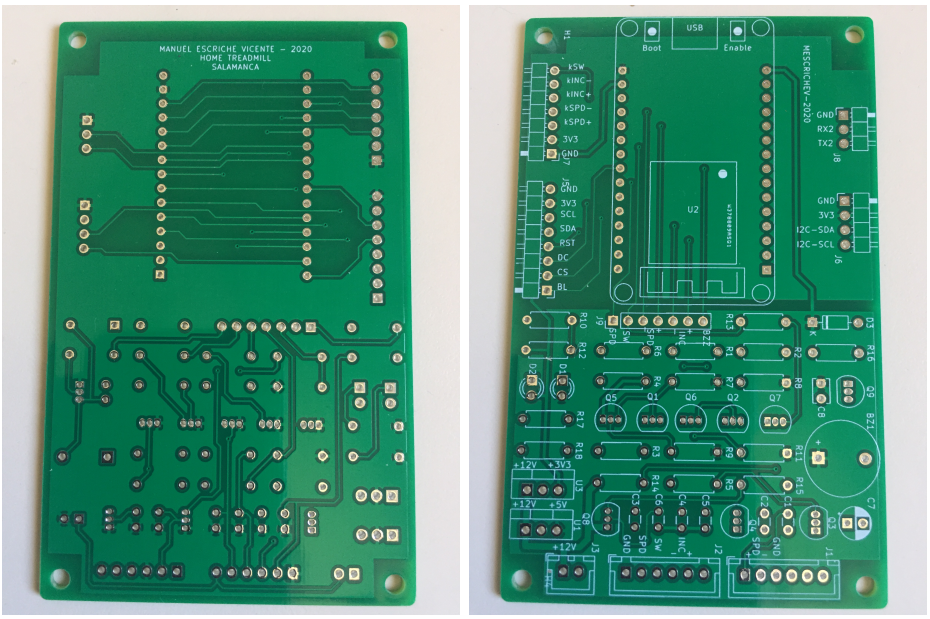

Figure 3: home treadmill pcb - implementation

3.5.1 Signals mapping

This is the final mapping between mcu pins and interfaces' signals.

TM Interface	
Signal	GPIO
INC+	27
INC-	19
SPD+	4
SPD-	2
SW	26
iSPD	33
BZZ	25

LCD Interface	
Signal	GPIO
BL	14
DC	32
RST	12
CS	5
SDA	23
SCL	18

Keypad Interface	
Signal	GPIO
SCL	15
SDA	13
INT	35

IIC Interface	
Signal	GPIO
SDA	21
SCL	22

RS232 Interface	
Signal	GPIO
RX2	16
TX2	17

Table 1: MCU - signals mapping

Obviously GPIOs are a way to point out what pins have been used. Strictly speaking GPIO(13,15) refers to I2C-0, GPIO(22,21) refers to I2C-1 interface, and GPIO(16,17) to UART2, and GPIO(5,18,23) to VSPI.

3.6 Drawbacks and Surprises

Well, not everything worked according to design and some modifications were introduced at hardware level in order to overcome specific drawbacks:

- GPIO1 and GPIO3 working as UART 0 as (TX,RX) were used by idf.py in order to monitor logs. So there was a conflict with using them for the signals (INC+, INC-) in the treadmill platform. I had to disconnect them in order to get logs.

- GPIO35 was used as an input for the INC+ input in the keypad, but GPIOs 34,35,36 and 39 have no way to configure any pull-up or pull-down resistor, as a consequence a lot of noise in the signal preventing its use.

- As first keypad prototype was set up soon it became evident 2 more keys were needed for speed. Four keys would allow to order decimal speed like 5.6 km/h. So instead of connecting key keypad directly to the MCU, the PCF8574 I/O Expander - I2C module was included.

Happily, adding more keys allowed to implement at software level speed programs without having to use a Raspberry Pi3, as initially planned.

3.7 Next Steps

Well, when we use the treadmill machine we all end up using the smart phone to play music.

1. Adding a bluetooth loadspeaker
2. Detach treadmill electronics into a module different from mcu would allow changing the controller.
3. A hardware module for dc power with usb connectors would allow adding bluetooth speakers.

4 Software

Well, the mcu has two cores: first one is used to control the speed motor, the other one to interact with the user.

Several RTOS tasks are created in order to attend all events happening while controlling motors.

- Buzzer task that produces beeps from orders sent to its queue.
- Screen task that displays blackboard information on the screen according to the program mode, and notifications.
- Keypad task that processes input information from keys through interruptions.
- Speed measurement task that works out speed from speed interruptions.
- Speed control task that controls motor speed to follow reference speed.
- Speed leader task that provides reference speed according to the speed program selected.
- Main task that supervises status, working time, and records speed race.

A blackboard is used to share information among tasks.

It's important to know the system behaves according to four modes:

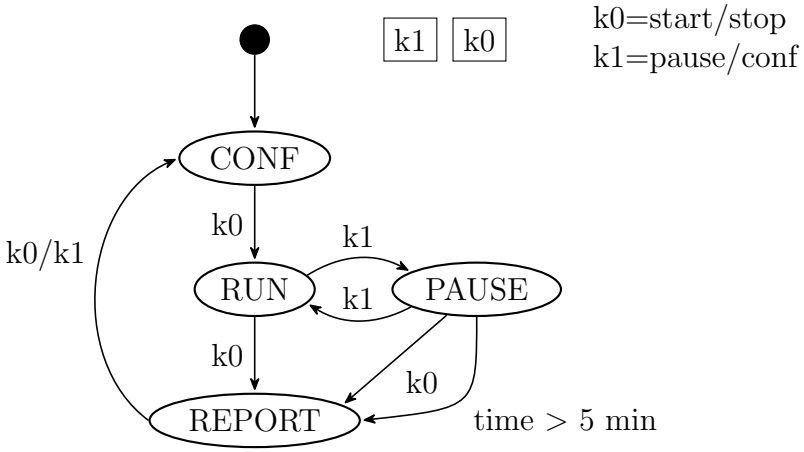

Figure 4: State diagram

CONF it refers to Configuration. It allows going into manual control where the user sets the reference speed and slope, and the speed program mode, where the user selects a specific speed program for the session.

RUN Once the start/stop key is pushed the motor runs according to reference speed. The key start/stop allows getting in and out.

PAUSE The pause key allows to temporarily stop the motor to take a breath, drink water, or anything. Maximum time in pause is 5 minutes. User get in and out by pushing on pause key.

REPORT Once stop key is pushed, the race report is shown on screen. By pushing start/stop or pause/conf keys you move on to Configuration in order to allow a new session.

Software has been structured in components:

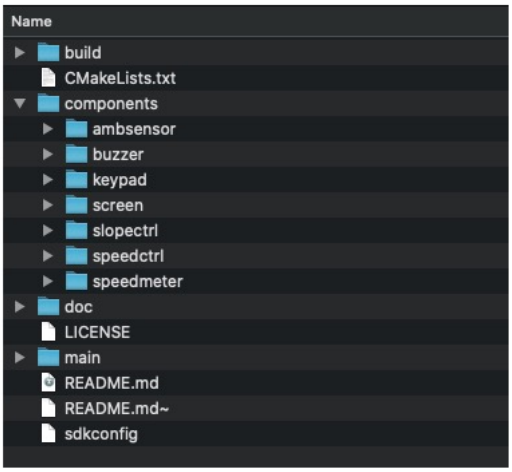

Figure 5: home treadmill software components

5 Figures and Tables

Contents

List of Figures

List of Tables