UNIVERSITÀ DEGLI STUDI DI UDINE
DIPARTIMENTO POLITECNICO DI INGEGNERIA
E ARCHITETTURA

Corso di Laurea Triennale in Ingegneria Elettronica

TESI DI LAUREA

PROGETTAZZIONE E SVILUPPO DI TIMER SU PIATTAFORMA IOT

Candidato
Riccardo Deana

Relatore
Pier Luca Montessoro

1 Introduzione

Chiunque abbia partecipato o assistito ad un esame universitario sa che la richiesta più frequente, durante lo svolgimento, è quella del tempo rimasto prima della consegna.
Un timer sarebbe utile per evitare di disturbare l'intera aula, dove sono presenti anche persone munite di orologio, e per non creare confusione nel caso siano presenti più esami in una sola stanza.
Questa tesi mira proprio a realizzare un timer per la gestione degli esami. Lo scopo è quello di proiettare i nomi delle prove che si svolgono in un'aula e il tempo rimanente prima della consegna di ognuna di esse.
Il progetto utilizza un single board computer, da collegare al proiettore tramite una porta HDMI. Questa è stata giudicata essere la soluzione migliore, considerando che nelle aule più grandi è sempre disponibile un proiettore e che l'Università di Udine ha a disposizione una grande quantità di single board computer. Una volta completato il progetto sarà quindi possibile realizzarne più esemplari da distribuire ai professori.
Sulla scheda è stato installato un sistema operativo basato su Linux, creato appositamente per questo tipo di dispositivi. Il programma utilizzato come timer è stato realizzato in C, per una maggiore ottimizzazione.
Il file di testo con i nomi degli esami e le relative durate viene letto da un'unità flash USB e presenta un sistema di commenti per ignorare una riga simile a quello del linguaggio C.
L'alternativa sarebbe stata leggere la lista degli esami da un file in rete ma sono stati trovati alcuni punti a sfavore di questa soluzione: la scheda non presenta una connessione WiFi, nelle aule la connessione cablata richiede l'autenticazione via browser e infine sarebbe stato più complicato per i docenti meno esperti.
Il conto alla rovescia parte quando viene premuto un pulsante fissato sulla scatola protettiva e collegato alla scheda. Avere un solo pulsante rende l'utilizzo del timer molto semplice e intuitivo.
Sono state prese le dovute precauzioni per evitare di danneggiare il filesystem quando la scheda viene scollegata dall'alimentazione mentre è ancora accesa. La chiavetta USB invece non corre rischi perché non vengono eseguite operazioni sui file presenti all'interno, dopo l'iniziale lettura della lista degli esami.

2 Piattaforma hardware

Il progetto si basa su un Orange Pi One, un single board computer prodotto da Shenzhen Xunlong Software CO, a partire dal 2016.
Le dimensioni di soli 69mm x 48mm e il peso di soli 36g rendono questa scheda facilmente trasportabile e quindi adatta allo scopo.
Il processore, basato sull'architettura ARM, è un Allwinner H3 Quad-Core Cortex-A7 e opera ad una frequenza di 1.2GHz.
La GPU è una Mali-400 MP2 a 600MHz, con supporto a OpenGL ES 2.0.
La RAM, di tipo DDR3 condivisa con la GPU, ammonta a 512MB.
Queste caratteristiche hardware permettono buone prestazioni con tutti i sistemi operativi testati. Infatti, per questa piattaforma, sono disponibili i porting di Android e dei più noti sistemi operativi basati sul Linux (Ubuntu, Debian, Fedora, OpenSuse, Arch Linux, Kali Linux). Inoltre sono disponibili sistemi operativi, quali Armbian e Raspbian, appositamente sviluppati per single board computer.
È presente uno slot MicroSD, che può ospitare una scheda di dimensione massima 32GB, dove è possibile installare il sistema operativo.

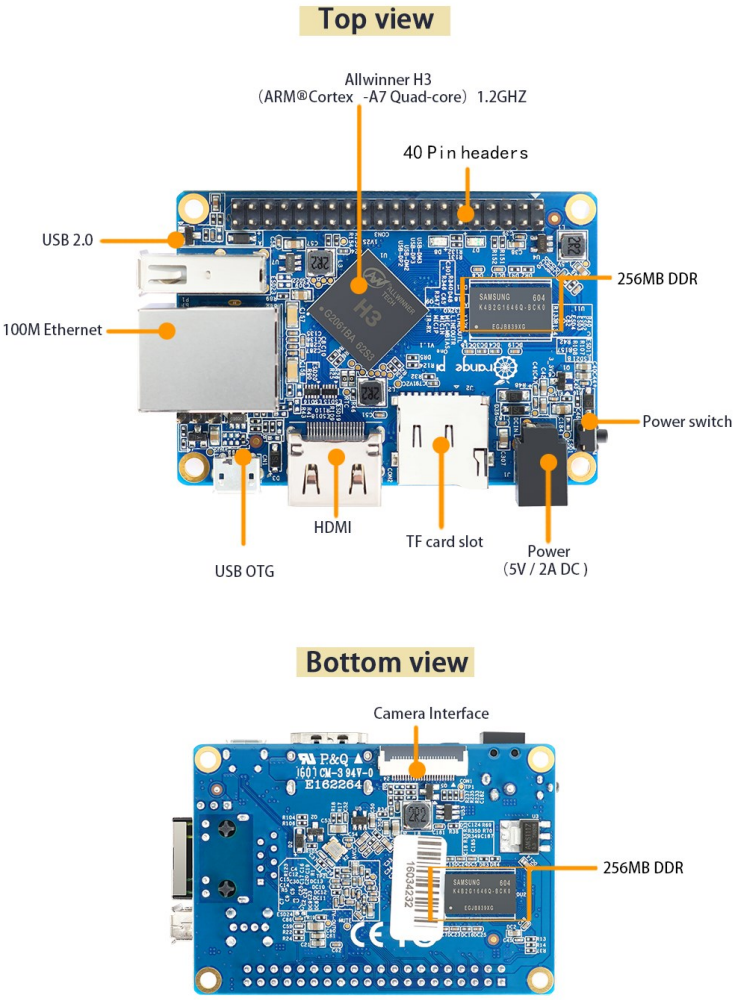


Figura 2.1: Aspetto di un Orange Pi One

L'uscita video è una HDMI Tipo A, con supporto al Consumer Electronics Control (CEC) e all'High-Bandwidth Digital Content Protection (HDCP). È possibile collegare delle periferiche alla scheda, tramite una porta USB 2.0 Tipo A e una porta Micro-USB OTG 2.0.
Il computer dispone di un connettore RJ45, che supporta connessioni fino a 100Mbps, utile per collegarlo alla rete internet e configurarlo.
Sulla scheda trovano posto anche 40 pin General Purpose Input/Output (GPIO), disposti in modo da essere compatibili con il Raspberry Pi modello B+ (figura 2.2).
Sul retro è installata una porta Camera Service Interface (CSI), che consente di collegare un ingresso video, come una telecamera, ad una risoluzione massima di 1080p a 30fps.
L'alimentazione a 5v è garantita da un connettore 4x1.7, che assorbe una corrente massima di 2A.

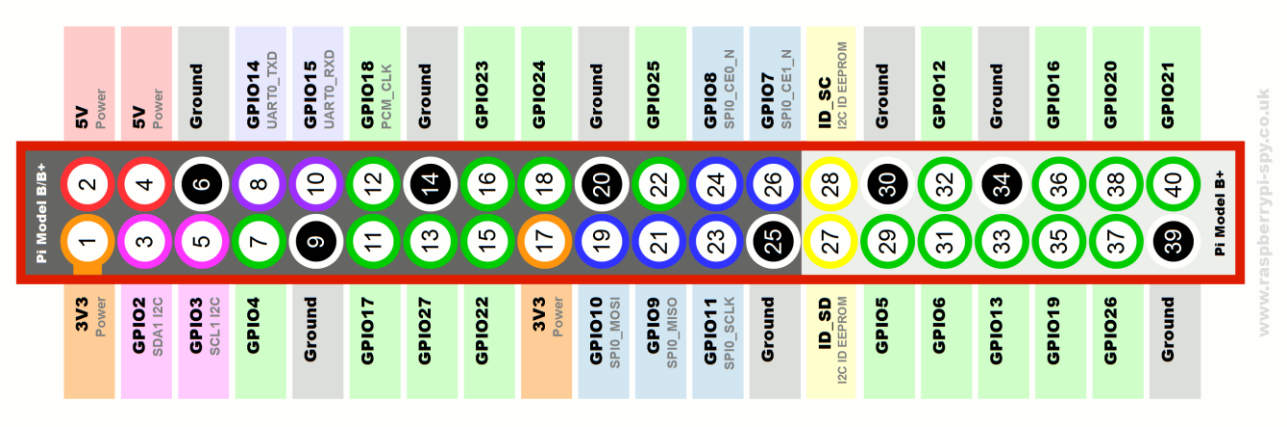


Figura 2.2: Disposizione dei pin in un Orange Pi One

3 Sistema operativo

Figura 3.1: Logo di Armbian

Come sistema operativo è stato scelto Armbian, in quanto è attualmente il più aggiornato disponibile per Orange Pi One.
Armbian è un sistema operativo molto leggero, sviluppato appositamente per single-board computer con architettura ARM, basato su Ubuntu o su Debian. È stata scelta la versione basata su Ubuntu, per il maggiore supporto disponibile in rete.
Al momento della scrittura di questa tesi l'ultima versione di Armbian disponibile è la 5.65, basata su Ubuntu Bionic Beaver 14.04 LTS con kernel Linux 4.14.
Il sistema viene scritto direttamente sulla scheda SD, utilizzando un PC con il programma Etcher, che esegue in modo semplice e automatico l'operazione.

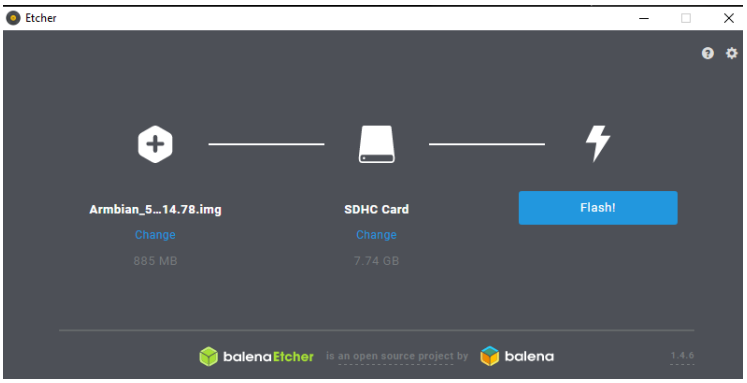


Figura 3.2: Etcher

4 Configurazione

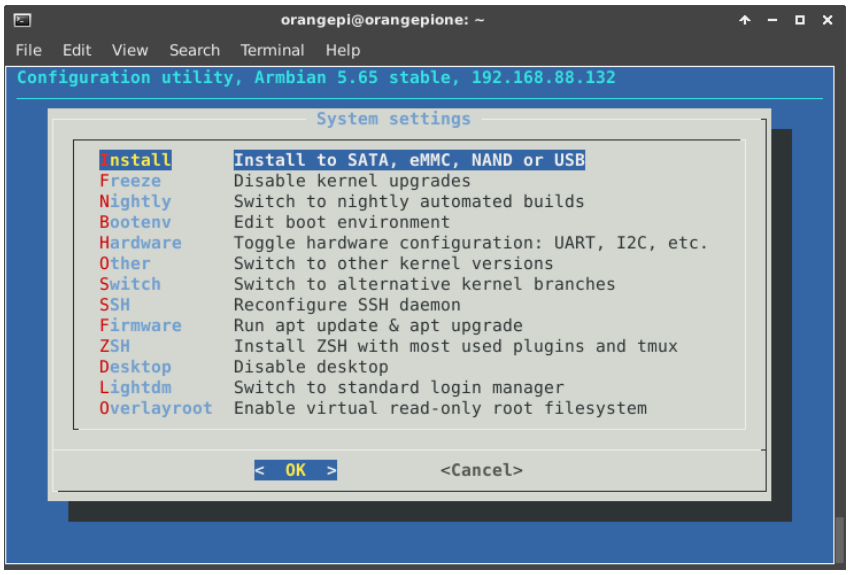


Figura 4.1: Armbian config

Al primo avvio il sistema è privo di interfaccia grafica e, tramite il terminale, richiede all'utilizzatore di cambiare la password di root e di creare un nuovo utente. In seguito è consigliabile collegare il dispositivo alla rete tramite un cavo ethernet ed eseguire il comando "sudo apt update" per aggiornare la lista dei pacchetti disponibili. Terminata questa operazione, è possibile eseguire "sudo armbian-config" per aprire una minimale interfaccia che consente, andando in System→Default, di installare l'ambiente desktop Xfce e alcuni programmi utili. Dopo il riavvio troviamo il sistema completo di interfaccia grafica e provvisto di tutti i pacchetti necessari per compilare ed eseguire programmi in C ed in Python.
Tramite il comando da terminale "sudo apt-get install libgtk-3-dev", è stato necessario installare gtk, un toolkit per la creazione di interfacce grafiche, che comprende anche la libreria per C.
Per questo progetto il puntatore del mouse è inutile e esteticamente sgradevole, quindi si è deciso di renderlo invisibile. A questo scopo è stato installato unclutter tramite il comando da terminale "sudo apt install unclutter". Questo programma fa sparire automaticamente il puntatore del mouse dopo un secondo di inattività.

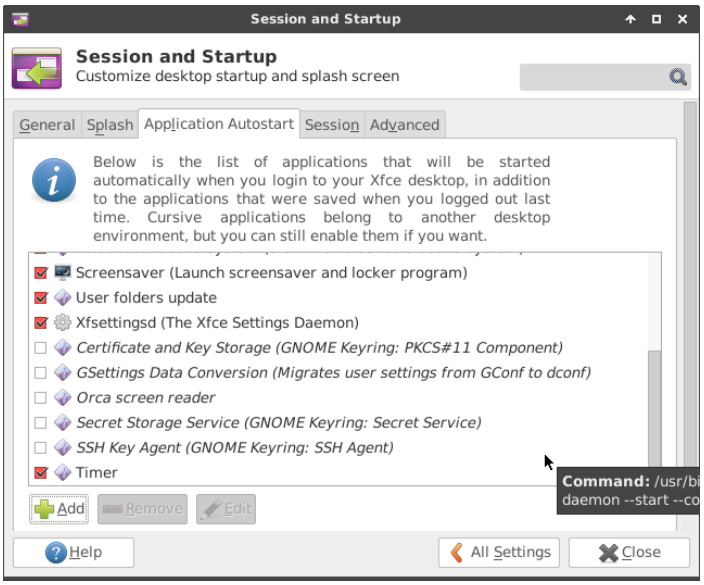
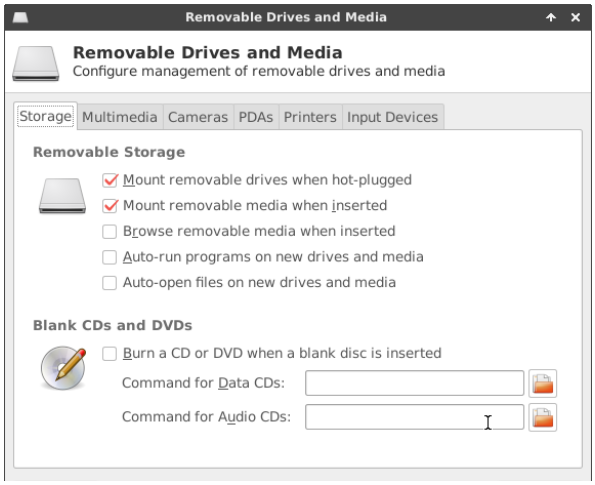


Figura 4.2: Session and Startup

Nelle impostazioni del sistema, alla voce "Session and Startup" (figura 4.2) è possibile aggiunge programmi da far partire all'avvio del sistema. In questo caso è stato aggiunto il programma in C chiamato timer.



È importante verificare che, nelle impostazioni alla voce "Removable Drives and Media" (figura 4.3), sia selezionata l'opzione di montare automaticamente i dispositivi removibili quando questo vengono collegati. Questa opzione però non garantisce che i dispositivi vengano montati automaticamente all'avvio.

Figura 4.3: Removable Drives and Media

È possibile eseguire dei comandi all'avvio del sistema anche aggiungendoli al file rc.local nella cartella /etc/. Dato che in raspbian i dispositivi USB non vengono automaticamente montati all'avvio è stato necessario aggiungere i comandi in figura 4.4 al suddetto file.
Il comando "sudo mkdir -p /media/orangepi/zzzzzz" crea, con privilegi di amministratore, la cartella zzzzzz in /media/orangepi/. Il tag -p serve ad evitare un errore, nel caso la cartella esista già.
In seguito si cerca di montare la prima partizione del dispositivo sdb nella cartella appena creata, in caso di fallimento si prova con il dispositivo sda e, qualora falliscano entrambi, si cancella la cartella. Normalmente il primo dispositivo di archiviazione USB collegato viene riconosciuto dal sistema come sda ma, per maggiore affidabilità, si è scelto di provare a montare prima sdb. È stato scelto il nome zzzzzz per la cartella, perché non insorgano problemi con il programma in C, nel caso in cui l'utente finale scolleghi e ricolleghi la chiavetta USB dopo l'avvio del sistema. Infatti, dopo l'avvio, il sistema monta automaticamente i dispositivi USB creando in /media/orangepi/ una cartella con il nome della chiavetta. In questo caso la cartella zzzzzz non viene eliminata ma la cartella creata automaticamente dal sistema sperabilmente la precede in ordine alfabetico. Questo fatto è utile in quanto il programma in C è stato pensato per cercare il file con la lista degli esami dentro la prima cartella all'interno di /media/orangepi/.

```
6  sudo mkdir -p /media/orangepi/zzzzzz
7  if ! sudo mount /dev/sdb1 /media/orangepi/zzzzzz; then
8    if ! sudo mount /dev/sda1 /media/orangepi/zzzzzz; then
9      sudo rm -rf /media/orangepi/zzzzzz
10   fi
11 fi
```

Figura 4.4: Montaggio della chiavetta

Per concludere stato necessario aggiungere al file rc.local, con privilegi di amministratore, anche il programma in Python chiamato button.py, che verrà illustrato nel prossimo capitolo.
A progetto terminato è possibile attivare l'Overlayroot, che rende il filesystem virtualmente di sola lettura e quindi evita che il sistema venga danneggiato quando la scheda viene spenta brutalmente. Se si vuole modificare lo stato di questa protezione, è sufficiente eseguire nuovamente "sudo armbian-config" nel terminale e spostarsi in System→ Overlayroot.

5 Collegamento del pulsante

Per avviare il timer era necessario un pulsante, che è stato collegato tra il pin 1, corrispondente alla tensione 3.3V, e il pin 7, corrispondente al GPIO4 (vedi figura 2.2).
Per gestire l'input, la soluzione più semplice e funzionale è stata scrivere un breve programma in Python, che è stato chiamato button.py (figura 5.1).
È stato necessario installare le librerie pyA20 e keyboard tramite il comando "pip install" da terminale.
La libreria pyA20 serve a gestire i pin. Purtroppo in questa libreria i pin sono chiamati con nomi diversi da quelli standard in figura 2.2 ma, seguendo le indicazioni sul sito dello sviluppatore, risulta facile capire la corrispondenza.

```
1   import time
2   from pyA20.gpio import gpio
3   from pyA20.gpio import port
4   import keyboard
5
6   gpio.init()  #inizializzo i pin
7
8   gpio.setcfg(port.PA6, gpio.INPUT)  #imposto il pin 7 come input
9
10  gpio.pullup(port.PA6, gpio.PULLDOWN)  #collego il pin 7 a massa tramite una
11                                        #resistenza di pulldown interna alla scheda
12
13  keyboard.press_and_release('a')  #la libreria ha un bug:
14                                   #la prima volta che questa funzione
15                                   #viene chiamata, non fa niente
16
17  while True:
18      if gpio.input(port.PA6) == 1:          #se il pin viene collegato a 3.3V
19          keyboard.press_and_release('a')    #simula la pressione del tasto a
20          time.sleep(0.1)                    #aspetta per 0.1s
21      time.sleep(0.1)
```

Figura 5.1: button.py

La funzione init serve a inizializzare i pin e va chiamata prima di utilizzare le altre funzioni della libreria.
Tramite la funzione setcfg è possibile impostare un determinato pin come input o come output.
La funzione pullup consente di collegare un pin a massa o ai 3.3V, tramite una resistenza da circa 10kΩ interna alla scheda. In questo modo la corrente massima, quando il pulsante viene premuto, è abbondantemente inferiore ai 16mA, erogabili al massimo da un pin. La funzione input legge lo stato del pin in modo digitale, distinguendo tra 0 e 1.
La funzione press_and_release della libreria keyboard consente di simulare la pressione di un tasto sulla tastiera. Sfortunatamente questa funzione ha un bug, che non fa eseguire il comando la prima volta che viene chiamata. È quindi necessario chiamarla una volta prima di iniziare il ciclo che controlla lo stato del pin. Successivamente si simula la pressione del tasto A, quando viene rilevata la pressione del pulsante.
La funzione sleep serve chiaramente a fermare l'esecuzione del programma per il tempo specificato.

6 Codice in C

Come prima cosa sono state incluse le librerie necessarie. Oltre alla libreria standard stdlib.h sono state incluse la libreria stdio.h per la gestione dell'input/output, la libreria string.h per la manipolazione delle stringhe, la libreria ctype.h per la classificazione dei caratteri, la libreria gtk.h per creare l'interfaccia grafica e infine la libreria time.h per le funzioni legate all'orario.

```c
#include <stdio.h>      //librerie
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <gtk/gtk.h>
#include <time.h>
```

```c


#define MAX_EXAMS 8     //massimo numero di esami che possono essere visualizzati
#define MAX_LINE 160    //massimo numero di caratteri in una riga del file di configurazione
#define MAX_NAME 140    //massimo numero di caratteri per il nome di un esame

```

In seguito sono stati definiti il massimo numero di esami che possono essere visualizzati (MAX_EXAMS), il massimo numero di caratteri in una riga del file di configurazione (MAX_LINE) e il massimo numero di caratteri per il nome di un esame (MAX_NAME).

```c
struct exam {
    char name[MAX_NAME];    //nome dell'esame
    int minutes;            //durata in minuti
    GtkLabel *nameLabel;    //casella di testo corrispondente al nome dell'esame nell'interfaccia grafica
    GtkLabel *timeLabel;    //casella di testo corrispondente al tempo rimanente nell'interfaccia grafica
    int blink;              //serve a far lampeggiare il testo "CONSEGNA"
};
```

Si è scelto di creare una struttura chiamata exam per agevolare la scrittura del codice. All'interno della struttura troviamo la stringa name con il nome dell'esame, l'intero minutes che corrisponde alla sua durata in minuti, le caselle di testo nameLabel e timeLabel, corrispondenti al nome dell'esame e al tempo rimanente nell'interfaccia grafica. Notiamo infine la presenza della variabile intera blink, che sarà utile in seguito per far lampeggiare il testo "CONSEGNA" alla fine dell'esame.

```c
struct exam exams[MAX_EXAMS];   //vettore che contiene tutti gli esami validi trovati
int examsCount = 0;             //numero di esami validi trovati
int width = 1920;               //larghezza dello schermo in pixel
int height = 1080;              //altezza dello schermo in pixel
time_t start;                   //ora di inizio degli esami
gboolean isStarted = FALSE;     //tiene conto del fatto che gli esami siano iniziati
PangoAttrList *attrlistName;    //attributi del testo
PangoAttrList *attrlistTime;
gboolean isConnected = FALSE;   //segnala se la chiavetta USB è collegata
```

Proseguendo nell'analisi del codice, troviamo le variabili globali.
Viene definito il vettore exams di strutture exam, che conterrà tutti gli esami validi. L'intero examsCount, inizializzato a 0, tiene conto del numero di esami validi trovati.

Gli interi width e height sono utili per memorizzare la risoluzione dello schermo e sono inizializzati in modo da corrispondere ad un monitor Full HD.
La variabile start di tipo time_t serve ad immagazzinare l'ora di inizio del conto alla rovescia. La variabile isStarted serve a verificare se gli esami sono iniziati.
AttrlistName e attrlistTime sono le liste degli attributi del testo utilizzato nell'interfaccia.
La variabile isConnected segnala se la chiavetta USB è collegata.

```
34  void setAttributes(void);              //prototipi delle funzioni
35  void createStartInterface(void);
36  gboolean readFile(void);
37  void createInterface(void);
38  gboolean set_exam(gpointer data);
39  void key pressed(GtkWindow *window);
```

Subito dopo sono presenti i prototipi delle funzioni, che verranno in seguito descritte.

```
42  int main(void){              //main
43
44      setAttributes();
45
46      createStartInterface();
47
48      createInterface();
49
50      return 0;
```

Il main è molto minimale, si limita a chiamare tre funzioni. Si è fatta questa scelta per rendere il codice ordinato e facilmente comprensibile.

6.1 La funzione setAttributes

La prima funzione ad essere chiamata è settAttributes, che imposta gli attributi del testo e rileva la risoluzione dello schermo.

```
55  void setAttributes(void){  //imposta gli attributi del testo e rileva la risoluzione dello schermo
56
57      int nameSize;
58      int timeSize;
59      GdkScreen *screen;
60      gtk_init(NULL, NULL);
61
62      if((screen = gdk_screen_get_default()) != NULL){   //rilevo l'altezza e la larghezza dello schermo in pixel
63          width = gdk_screen_get_width(screen);
64          height = gdk_screen_get_height(screen);
65      }
66
67      nameSize = (25000*height)/1080;  //l'interfaccia è stata tarata su uno schermo FullHD
68      timeSize = (40000*height)/1080;  //viene fatta la proporzione per adattarla ad ogni schermo
```

Gli interi nameSize e timeSize sono le due dimensioni del testo utilizzate.
La funzione gtk_init va chiamata prima di iniziare ad utilizzare le altre funzioni della libreria gtk.
Se il rilevamento dello schermo attraverso gdk_screen_get_default va a buon fine, le informazioni vengono immagazzinate nella variabile screen. Le funzioni gdk_screen_get_width e gdk_screen_get_height servono a trovare la larghezza e l'altezza dello schermo in pixel. In seguito viene fata una proporzione per adattare la dimensione del testo, tarata su uno schermo Full HD, ad ogni risoluzione.

```
70      PangoAttribute *attrName;                        //utilizzo un font più piccolo per il nome degli esami
71      attrlistName = pango_attr_list_new();
72      attrName = pango_attr_size_new (nameSize);
73      pango_attr_list_insert(attrlistName, attrName);
74
75      PangoAttribute *attrTime;                       //ed uno più grande per il tempo rimasto e i messaggi di errore
76      attrlistTime = pango_attr_list_new();
77      attrTime = pango_attr_size_new (timeSize);
78      pango_attr_list_insert(attrlistTime, attrTime);
79
80  }
```

Vengono definiti gli attributi del testo, inizializzati e inseriti nelle rispettive liste. Si utilizzano due dimensioni per il testo: una più piccola per il nome degli esami ed una più grande per il tempo rimasto ed i messaggi di errore.

6.2 La funzione createStartInterface

La funzione createStartInterface visualizza l'interfaccia iniziale, mentre il programma cerca la chiavetta con il file con la lista degli esami.

```
82
83  void createStartInterface(void){  //visualizza l'interfaccia iniziale
84
85      GtkWindow *window;
86      GtkLabel *startLabel;
87
88      gtk_init(NULL, NULL);  //inizializzo gtk
89
90      window = gtk_window_new(GTK_WINDOW_TOPLEVEL);  //creo una nuova finestra
91      gtk_window_fullscreen (window);                //faccio in modo che sia visualizzata a pieno schermo
92      g_signal_connect (window, "destroy", G_CALLBACK (gtk_main_quit), NULL);
93
94      startLabel = gtk_label_new("Ricerca di dispositivi USB...");   //creo una nuova casella di testo
95      gtk_label_set_attributes(startLabel, attrlistTime);            //imposto gli attributi del testo
```

Vengono definite una finestra e una casella di testo.
La finestra viene creata con la funzione gtk_window_new e impostata a schermo intero con la funzione gtk_window_fullscreen. La funzione g_signal_connect chiama la funzione gtk_main_quit, che termina il main loop di gtk, nel caso la finestra venga chiusa per qualche motivo. Chiaramente il programma non è pensato perché la finestra venga chiusa ma, per sicurezza, è meglio prevedere questa eventualità.
La funzione gtk_label_new crea una nuova casella di testo con la scritta "Ricerca di dispositivi USB...". Successivamente vengono impostati gli attributi del testo.

```
96      gtk_label_set_ellipsize(startLabel, PANGO_ELLIPSIZE_END);      //accorcia il testo con dei punti
97                                                                     //qualora non dovesse esserci abbastanza spazio
98      gtk_label_set_xalign(startLabel, 0.5);                         //centro il testo
99      gtk_container_add(GTK_CONTAINER(window), startLabel);          //inserisco la casella di testo nella finestra
100
101     gtk_widget_show_all(window);                                   //visualizzo la finestra
102
103     g_timeout_add(2000,readFile, NULL);  //chiamo la funzione che legge il file con la lista degli esami dopo 2s
104
105     gtk_main();   //inizio il main loop di gtk
106
107 }
```

La funzione gtk_label_set_ellipsize accorcia il testo con dei puntini alla fine, nel caso non ci sia abbastanza spazio. La funzione gtl_label_set_xalign, con argomento 0.5, centra il testo nella casella. Gtk_container_add inserisce la casella di testo nella finestra.
La funzione g_timeout_add chiama la funzione che legge il file con la lista degli esami dopo 2 secondi dall'inizio dei main loop.
La funzione gtk_main fa iniziare il main loop di gtk, visualizzando l'interfaccia.

6.3 La funzione readFile

La funzione readFile cerca la chiavetta e, nel caso sia collegata, legge il file con la lista degli esami.

```
110  gboolean readFile(void){  //legge il file con la lista degli esami
111
112      char line[MAX_LINE];
113      char *name;
114      char *hours;
115      char *minutes;
116      int time;
117      FILE *config;
118      int i;
119
120      char *user = getenv("USER");  //legge il nome utente dal sistema
121      char path[150];               //il percorso della chiavetta USB
122      struct dirent *de;
123      int attempts = 0;             //numero di tentativi di trovare un dispositivo USB
124
125      strcpy(path, "/media/");
126      strcat(path, user);
127      DIR *dr = opendir(path);      //apre la cartella
```

Viene definita la stringa line che corrisponde ad una linea del file di configurazione. In seguito la stringa viene divisa nelle tre stringhe name, hours e minutes.
La funzione getenv(“USER”) acquisisce dal sistema il nome utente. Chiaramente il programma sarà avviato sempre dallo stesso utente, quindi questa operazione potrebbe sembrare superflua ma rende il programma compatibile con tutti i dispositivi, con un costo computazionale trascurabile. La stringa path rappresenta il percorso della chiavetta USB mente l’intero attempts tiene conto dei tentativi di trovare un dispositivo USB.
La chiavetta viene montata in una cartella contenuta in /media/NomeUtente/.
La funzione opendir(path) apre la cartella specificata dalla stringa path.

```
129      while ((attempts < 80) && !(isConnected)){  //se non è connessa una chiavetta USB
130
131          if ((de = readdir(dr)) == NULL){
132              rewinddir(dr);    //riparte dall'inizio della cartella
133              attempts++;
134              usleep(500000);   //attende 500ms prima di riprovare
135          }else if ((de->d_name[0] != '.')){        //se la cartella non è vuota
136              isConnected = TRUE;
137          }
138
139      }
```

Fino a che non è stata connessa una chiavetta USB e fino a che non si sono fatti 80 tentativi, si cerca di leggere la cartella, ripartendo sempre dall’inizio e si aumentata il numero di tentativi. Se la cartella /media/NomeUtente/ non è vuota viene segnalata la connessione della chiavetta.
80 tentativi corrispondono circa a 40s, dato che si attendono 500ms con la funzione usleep prima di riprovare.

```
141     if ((attempts < 80) && (de != NULL)){
142         strcat(path, "/");
143         strcat(path, de->d_name);
144         strcat(path, "/esami.txt");  //ricostruisco il percorso completo del file con la lista degli esami
145 
146         if((config = fopen(path, "r")) == NULL){  //se non riesco ad aprire il file
147             examsCount = -1;  //non è stato trovato il file con la lista degli esami
148         }else{
149 
150             while((!feof(config)) && (examsCount < MAX_EXAMS)){  //leggo il file fino a che non arrivo alla fine
151 
152                 fgets(line,MAX_LINE,config);     //leggo una riga del file
```

Se sono stati fatti meno di 80 tentativi, viene ricostruito, tramite la funzione strcat, il percorso del file esami.txt, nel quale è contenuta la lista degli esami.
Se il programma non riesce ad aprire il file, segala che non è stato trovato alcun file di configurazione ponendo examsCount a -1.
Altrimenti il file viene letto una riga alla volta fino alla fine o fino a che non si è raggiunto il numero massimo di esami.

```
154             i = 0;
155             while(line[i] == ' '){  //conto gli spazi all'inizio della riga
156                 i++;
157             }
158 
159             if(line[i] == '/' && name[i+1] == '/'){  //se trovo i simboli // ignoro la riga
160                 time = 0;
161             }else{
162                 name = strtok(line,"=");         //il nome dell'esame è a sinistra del simbolo =
163                 hours = strtok(NULL,".,:\n\r");  //le ore sono a sinistra del simbolo . oppure : oppure a capo
164                 minutes = strtok(NULL,"\n\r");   //i minuti sono a sinistra del simbolo a capo
165                 if((hours != NULL) && (minutes != NULL)){         //converto le stringhe minutes e hours in un intero che rappresenta i minuti di durata
166                     time = 60 * atoi(hours) + atoi(minutes);
167                 }else if((hours != NULL) && (minutes == NULL)){
168                     time = 60 * atoi(hours);
169                 }else{
170                     time = 0;
171                 }
172             }
```

Utilizzando l'intero i vengono contati gli spazi presenti all'inizio della riga.
Era necessario introdurre un sistema di commenti simile a quello del linguaggio C, per riuscire a far ignorare al programma una riga del file. Quindi, se vengono trovati i simboli //, la riga viene ignorata ponendo la durata time a 0.
Se la riga non viene ignorata, deve essere scomposta in: nome dell'esame, che si trova a sinistra del simbolo "=", ore, che sono a sinistra del simbolo "." oppure ":" oppure "a capo" e infine minuti che sono a sinistra del simbolo "a capo".
In seguito viene fatta, attraverso la funzione atoi, la conversione in interi delle stringhe precedentemente divise.

```
174             if(time != 0){
175 
176                 for (i=0; i<strlen(name); i++){
177                     name[i] = toupper(name[i]);          //scrivo il nome in maiuscolo
178                 }
179 
180                 strcpy(exams[examsCount].name, name);  //popolo il vettore di struct exam con
181                 exams[examsCount].minutes = time;      //le informazioni lette dal file
182                 exams[examsCount].blink = 0;           //inizializzo blink a 0
183                 examsCount++;
184 
185             }
```

Se la riga del file non presenta problemi, viene popolato il vettore di struct exam con le informazioni lette e viene aumentati il numero di esami validi trovati

Prima di inserirlo nel vettore, il nome del file viene reso maiuscolo con la funzione toupper, per renderlo più leggibile.

```
189             fclose(config);  //chiudo il file
190         }
191     }else{
192         examsCount = -2;  //non è stata trovata una chiavetta USB
193     }
194
195     closedir(dr);     //chiudo la cartella
196
197     gtk_main_quit();  //esco dal main loop di gtk
198
199     return FALSE;     //la funzione non verrà più chiamata
200
201 }
```

Il file viene chiuso con la funzione fclose. Se non è stata trovata alcuna chiavetta USB, viene segnalato ponendo examsCount a -2. Con la funzione closedir si chiude la cartella e con gtk_main_quint si esce dal main loop di gtk.
Infine la funzione readFile ritorna FALSE in modo da non essere più chiamata.

6.4 La funzione createInterface

La funzione createInterface visualizza l'interfaccia principale del programma, con i nomi degli esami e il tempo rimanente.

```
204  void createInterface(void){   //visualizza l'interfaccia principale
205
206      GtkWindow *window;
207      GtkWidget *grid;
208
209      int e = 0;
210
211      int margin;
212      margin = (80*width)/1920;  //definisco il margine e lo scalo in base alla larghezza dello schermo
213
214      gtk_init(NULL, NULL);   //inizializzo gtk
215
216      window = gtk_window_new(GTK_WINDOW_TOPLEVEL);   //creo una nuova finestra
217      gtk_window_fullscreen (window);                 //faccio in modo che sia visualizzata a pieno schermo
218      g_signal_connect (window, "destroy", G_CALLBACK (gtk_main_quit), NULL);
219
220      grid = gtk_grid_new ();                          //creo una nuova griglia
221      gtk_grid_set_row_homogeneous (grid, TRUE);       //le righe devono avere tutte la stessa altezza
222      gtk_grid_set_column_homogeneous (grid, TRUE);    //le colonne devono avere tutte la stessa larghezza
223      gtk_widget_set_margin_end(grid, margin);         //imposto i margini destro e sinistro
224      gtk_widget_set_margin_start(grid, margin);
225      gtk_container_add(GTK_CONTAINER(window), grid);   //inserisco la griglia nella finestra
```

Viene definita la dimensione del margine e viene calcolata facendo una proporzione sulla larghezza dello schermo.
La finestra window viene creata e resa full screen.
In seguito viene creata la griglia grid, le cui righe e colonne vengono rese omogenee, ovvero tutte della stessa dimensione.
Si impongono i margini destro e sinistro uguali al valore calcolato in precedenza. Per ultimo la griglia viene inserita all'interno della finestra.

```
227      if (examsCount > 0){  //se è stato trovato almeno un esame valido
228
229          for (e = 0; e < examsCount; e++){  //scorro il vettore degli esami
230
231              exams[e].nameLabel = gtk_label_new(exams[e].name);   //creo i campi di testo per il nome e il tempo rimasto
232              exams[e].timeLabel = gtk_label_new(NULL);
233
234              gtk_label_set_ellipsize(exams[e].nameLabel, PANGO_ELLIPSIZE_END);   //accorcia il testo con dei punti
235              gtk_label_set_ellipsize(exams[e].timeLabel, PANGO_ELLIPSIZE_END);   //qualora non dovesse esserci abbastanza spazio
236
237
238              gtk_label_set_xalign(exams[e].nameLabel, 0);  //allineo il nome a sinistra
239              gtk_label_set_xalign(exams[e].timeLabel, 1);  //allineo il tempo rimanente a destra
240
241              gtk_label_set_attributes(exams[e].nameLabel, attrlistName);  //imposto gli attributi del testo
242              gtk_label_set_attributes(exams[e].timeLabel, attrlistTime);
243
244              gtk_grid_attach (grid, exams[e].nameLabel, 1, e, 3, 1);  //inserisco la casella di testo del nome nella griglia alla colonna 1,
245                                                                       //riga e, con larghezza 3 colonne e altezza 1
246              gtk_grid_attach (grid, exams[e].timeLabel, 4, e, 1, 1);  //inserisco la casella di testo del tempo rimanente alla colonna 4, riga e
247
248              set_exam(e);  //chiamo la funzione che riempie le caselle di testo e le passo il numero dell'esame
249
250          }
```

Se è stato trovato almeno un esame valido, si scorre il vettore degli esami per creare i campi di testo per il nome dell'esame e il tempo rimasto. La casella di testo con il nome dell'esame viene già scritta e non verrà più modificata. Come di consueto, se non c'è abbastanza spazio, il testo viene accorciato usando dei puntini alla fine per evitare problemi con l'interfaccia.

Il nome dell'esame viene allineato a sinistra, mentre il tempo rimanete è allineato a destra.
Le caselle di testo vengono poi inserite all'interno della griglia nelle posizioni corrette.
Infine viene chiamata la funzione set_exam che riempie le caselle di testo del tempo rimanente.

```
252     }else{  //se non è stato trovato alcun esame valido
253
254         GtkLabel *errorLabel;  //creo una casella di testo
255
256         if(examsCount == 0){  //capisco il tipo di problema riscontrato
257             errorLabel = gtk_label_new("Non è stato trovato alcun esame valido");
258         }else if(examsCount == -1){
259             errorLabel = gtk_label_new("Non è stato trovato alcun file di configurazione");
260         }else if(examsCount == -2){
261             errorLabel = gtk_label_new("Non è stato trovato alcun dispositivo USB");
262         }
263
264         gtk_label_set_xalign(errorLabel, 0.5);                      //centro il testo
265         gtk_label_set_ellipsize(errorLabel, PANGO_ELLIPSIZE_END);   //accorcia il testo con dei punti, qualora non dovesse esserci abbastanza spazio
266         gtk_label_set_attributes(errorLabel, attrlistTime);         //imposto gli attributi del testo
267         gtk_grid_attach (grid, errorLabel, 1, 1, 1, 1);             //inserisco la casella di testo con l'errore nella griglia
268     }
```

Se non è stato trovato alcun esame valido viene creata una nuova casella di testo, nella quale scrivere il tipo di problema riscontrato.
Possono presentarsi tre tipi di errori: non è stato trovato alcun esame valido all'interno del file esami.txt, non è stato trovato alcun file con la lista degli esami all'interno della chiavetta oppure non è stata trovato alcun dispositivo USB. Per distinguere il tipo di problema il programma controlla il valore di examsCount.
Nella casella il testo viene centrato, accorciato, nel caso non ci sia abbastanza spazio e gli vengono assegnati gli attributi.
Poi la casella di testo viene inserita all'interno della griglia. In questo caso la griglia ha una sola riga e una sola colonna, quindi la casella di testo è a pieno schermo.

```
270     g_signal_connect (G_OBJECT (window), "key_press_event", G_CALLBACK (key_pressed), NULL);
271
272     gtk_widget_show_all(window);  //visualizzo la finestra
273
274     gtk_main();  //inizio il main loop di gtk
275
276 }
```

La funzione g_signal_connect fa in modo che, quando viene rilevata la pressione di un pulsante, venga chiamata la funzione key_pressed, che fa partire il timer.
Infine viene visualizzata la finestra con la funzoine gtk_widget_show_all e viene fatto iniziare il main loop di gtk.

6.5 La funzione set_exam

La funzione set_exam aggiorna la casella di testo del tempo rimasto corrispondente all'esame, il cui numero le viene passato tramite un puntatore.

```
279 gboolean set_exam(gpointer data){  //aggiorna le caselle di testo corrispondenti agli esami
280
281     int seconds;
282     char string [16];
283     int h;
284     int m;
285     int s;
286     time_t now;
287
288     int i = (int*)data;
289
290     time(&now);  //acquisisco l'ora attuale
291
292     seconds = exams[i].minutes * 60;  //converto i minuti in secondi
293
294     if (isStarted == TRUE){               //se gli esami sono iniziati
295       seconds -= difftime (now,start);    //faccio la differenza tra l'ora di partenza e quella attuale
296     }
297
298     h = (seconds/60)/60;  //conversione in ore, minuti e secondi
299     m = (seconds/60)%60;
300     s = (seconds%60)%60;
```

L'intero seconds corrisponde ai secondi totali di durata di un esame. Invece gli interi h, m, s, corrispondono alle ore, minuti e secondi rimanenti.
La variabile now di tipo time_t, serve ad immagazzinare l'ora attuale, che viene acquisita attraverso la funzione time.
L'intero i viene ricavato da un puntatore e serve ad identificare il numero dell'esame.
Se gli esami sono iniziati, viene fatta la differenza in secondi tra l'orario di partenza e quello attuale. In questo modo si evita che si accumulino errori dovuti all'imprecisione delle funzioni sleep o usleep, garantendo un calcolo dei secondi rimasti sempre preciso.
I secondi rimasti vengono convertiti in ore, minuti e secondi.

```
302     if ((h<=0) && (m<=0) && (s<=0)){  //se l'esame è finito
303       if(m > -2){  //se l'esame è finito da meno di 2 minuti
304         if(exams[i].blink < 7){
305           gtk_label_set_label(exams[i].timeLabel, "CONSEGNA");  //la scritta CONSEGNA lampeggia
306           exams[i].blink++;
307         }else if(exams[i].blink < 10){
308           gtk_label_set_label(exams[i].timeLabel, "        ");
309           exams[i].blink++;
310         }else{
311           exams[i].blink = 0;
312         }
313       }else{  //se l'esame è finito da più di 2 minuti
314         gtk_label_set_label(exams[i].timeLabel, "FINE");
315         return FALSE;  //la funzione non verrà più chiamata
316       }
317     }else{  //se l'esame non è finito
318       sprintf(string,"%02d:%02d:%02d", h, m, s );
319       gtk_label_set_label(exams[i].timeLabel, string);  //aggiorna il tempo rimanente nella casella di testo
320     }
321
322     return TRUE;  //la funzione verrà chiamata ancora
323
324 }
```

Se l'esame è finto da meno di due minuti, nella casella di testo del tempo rimanente lampeggia la scritta "CONSEGNA". Per far lampeggiare il testo, si cicla da 0 a 10 sulla variabile blink. Quando questa è compresa tra 0 e 6 il testo viene scritto, quando è compresa tra 7 e 10 il testo viene cancellato, scrivendo degli spazi.
La funzione gtk_label_set_label serve a scrivere una stringa in una casella di testo.
Se l'esame è finito da più di due minuti viene scritto "FINE" e la funzione ritorna FALSE, in modo da non essere più chiamata per quello specifico esame.
Se l'esame non è finito, con la funzione sprintf viene costruita la stringa e questa viene scritta nella casella di testo per aggiornare il tempo rimanente.
Infine viene ritornato TRUE per fare in modo che la funzione venga nuovamente chiamata.

6.6 La funzione key_pressed

La funzione key_pressed fa iniziare gli esami quando viene premuto il pulsante.

```
327 void key_pressed(GtkWindow *window){   //fa iniziare gli esami
328
329     int e = 0;
330
331     if(isStarted == FALSE){   //se gli esami non sono ancora iniziati
332         time(&start);          //acquisisco l'ora di partenza
333         isStarted = TRUE;      //gli esami sono iniziati
334
335         for (e = 0; e < examsCount; e++){  //scorro il vettore degli esami
336             g_timeout_add(100,set_exam, e);  //ogni 100ms chiamo la funzione che aggiorna le caselle di testo
337         }
338
339     }
340
341 }
342
```

Se gli esami non sono ancora iniziati, viene acquisita l'ora di partenza con la funzione time e viene segnalato l'inizio nella variabile isStarted.
Dopo viene scorso il vettore degli esami e, ogni 100ms, viene chiamata la funzione che aggiorna le caselle di testo del tempo rimanente. Alla funzione set_exam viene passato la posizione nel vettore dell'esame corrente.

7 Risultato finale

Per il docente è sufficiente preparare un file chiamato “esami.txt” con la lista dei suoi esami e delle relative durate. Il nome dell’esame deve essere seguito dal simbolo “=” e dalla durata in ore e minuti. Per separare le ore dai minuti si possono usare indistintamente i simboli “.” o “:”. Se dopo il simbolo di uguale è presente un solo numero, questo viene interpretato come una durata in ore. Il docente può commentare una riga, facendola iniziare con i simboli “//”; in questo modo il programma ignora quella specifica prova.
Una volta pronto, il file deve essere salvato su un dispositivo di archiviazione USB.

```
1  fondamenti di programmazione e strutture dati e algoritmi=3
2  Architettura dei calcolatori=2
3  reti di calcolatori = 1.30
4  sicurezza informatica = 1:30
5  //teoria delle reti elettriche ed elettrotecnica =0:01
6  //elaborazione numerica del segnale=   0.01
7  //teoria dei segnali e comunicazioni elettriche=0:45
8  //modellizzazione e controllo di sistemi dinamici=2.15
9
```

Figura 7.1: Esempio di file esami.txt

Per utilizzare il timer basta inserire la chiavetta USB e il cavo HDMI nell’Orange Pi e collegarlo in seguito all’alimentazione. Quando il dispositivo ha concluso l’avvio viene visualizzata la scritta “Ricerca di dispositivi USB…”, mentre il programma cerca la chiavetta con il file di testo contenente la lista degli esami.
Finita la ricerca del file di configurazione, il docente può premere il pulsante posto sulla scatola per far partire il conto alla rovescia.
Se il programma non trova il file o il dispositivo di archiviazione USB provvede ad avvisare l’utente con degli opportuni messaggi d’errore.

Figura 7.2: Aspetto del timer

Figura 7.3: Schermata iniziale

In seguito, se viene trovato il file con la lista degli esami, il programma presenta un'interfaccia come quella in figura 7.4, dove sono visualizzate contemporaneamente 6 diverse prove.

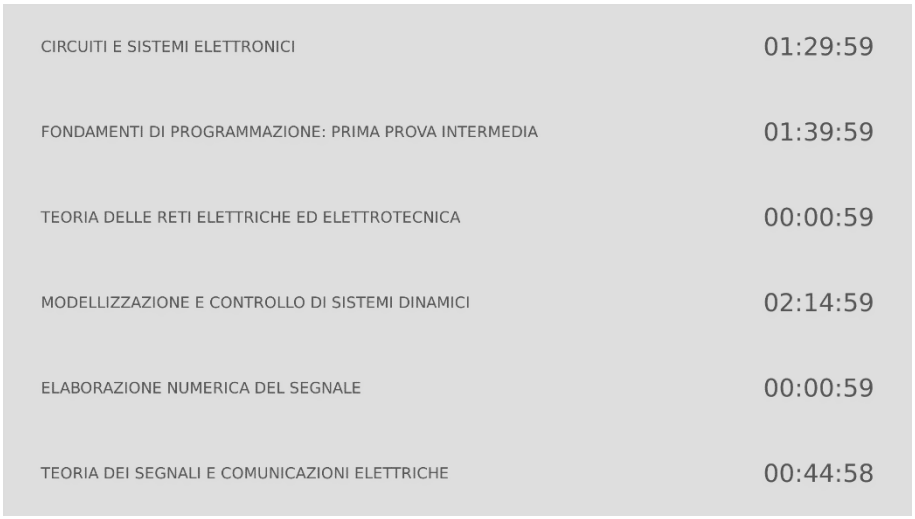


Figura 7.4: Timer in funzione

8 Conclusioni

Si è preferito mantenere l'interfaccia grafica più semplice possibile per rendere facilmente leggibile agli studenti la lista degli esami. A seconda dei gusti estetici, sarebbe possibile cambiare il colore dello sfondo o addirittura inserire un'immagine.
Disponendo di una versione diversa o più recente di single board computer con connettività WiFi sarebbe possibile ottenere la lista degli esami e le relative durate dalla rete. Con la connessione senza fili l'autenticazione sarebbe semplice, in quanto basterebbe impostare lo username e la password. Chiaramente si dovrebbe realizzare l'interfaccia per il caricamento del file in rete in modo da essere il più semplice possibile, affinché possa essere utilizzata anche dai docenti meno esperti. Comunque questa soluzione potrebbe essere disponibile in alternativa, senza eliminare la lettura del file dalla chiavetta USB.
Utilizzando single board computer più famosi e quindi supportati sarebbe possibile scrivere il programma in un linguaggio di livello più alto come Java. In questo caso non sarebbe necessario ricorrere a soluzioni creative per interfacciarsi con il pulsante e per montare la chiavetta USB. Infatti, per la realizzazione di progetti simili a questo, le schede più note dispongono di librerie funzionanti e prevedono alcune impostazioni utili direttamente nel sistema operativo.
Un altro sviluppo interessante sarebbe l'utilizzo di un modulo real time clock (RTC), da collegare alla scheda per mantenere l'orario memorizzato. Sarebbe quindi possibile proiettare anche l'ora attuale oltre al tempo rimanente. Questa soluzione è stata testata con successo ma presenta alcuni problemi: il cambio tra ora solare e ora legale non è automatico e richiederebbe un aggiornamento manuale dell'orario, l'università non dispone di moduli RTC per poter realizzare diversi esemplari del progetto, il modulo RTC richiede una batteria tampone aggiuntiva che può scaricarsi.

Riferimenti

http://www.orangepi.org/images/orangepione_info.jpg

http://www.orangepi.org/orangepione/

https://www.raspberrypi-spy.co.uk/wp-content/uploads/2012/06/Raspberry-Pi-GPIO-Layout-Model-B-Plus-rotated-2700x900.png

https://www.armbian.com/wp-content/uploads/2018/03/logo2.png

https://www.armbian.com/

https://docs.armbian.com/

https://www.balena.io/etcher/

https://pypi.org/project/pyA20/

https://pypi.org/project/keyboard/

http://manpages.ubuntu.com/manpages/bionic/man1/unclutter.1.html

https://www.gtk.org/

https://developer.gnome.org/

https://stackoverflow.com/

Paul Deitel Harvey Deitel "C Corso completo di programmazione" (Apogeo, 2010)

Ringraziamenti

Innanzitutto ringrazio il Professor Pier Luca Montessoro per avermi proposto questo interessante progetto e per avermi seguito e supportato assieme al collaboratore Giovanni Bortolin durante la sua realizzazione.
Una particolare ringraziamento a tutta la mia famiglia per essermi stata vicina in questi anni di studio e per avermi aiutato a superare tutti momenti difficili.
Vorrei ringraziare i miei amici per avermi offerto dei piacevoli momenti di svago.
Infine ci tengo a ringraziare personalmente il mio amico Luca, il cui sostegno è stato fondamentale.