

CENTRALINO

Di Castellani Filippo.

Data ultima modifica

7/7/2017

Dispositivo fisico CENTRALINO.

- Documentazione generale di progetto :
 - Idea del progetto.
 - Caratteristiche.
 - Funzioni.
 - Disegno forma e cianografia.
 - Interfaccia utente.
 - Elenco componenti.
 - Scelta componenti e preventivo.
 - Simulazioni.
- Scheda di progetto.
 - Analisi dei requisiti.
 - Analisi delle competenze necessarie.
 - Architettura di sistema.
 - Specifiche implementative.
- Documento tecnico :
 - Documentare il completamento del progetto e tabella sui tempi delle attività.
 - Strumenti utilizzati (Software).
 - Strumenti utilizzati (Hardware).
 - Codice.
 - Spiegazione del codice e le sue funzionalità e particolarità.
 - Problematiche degne di nota affrontate nella creazione del codice.

- Manutenzione.

Applicazione software CENTRALINO.

- Documentazione generale di progetto :
 - Idea del progetto
 - Caratteristiche.
 - Interfaccia utente.
 - Simulazioni.
 - Architettura di sistema.
 - Specifiche implementative.
- Documento tecnico :
 - Documentare il completamento del progetto e tabella sui tempi delle attività.
 - Codice.
 - Spiegazione del codice e le sue funzionalità e particolarità.
 - Problematiche degne di nota affrontate nella creazione del codice.
 - Manutenzione.
- Licenze
- Codici e relative spiegazioni.
 - Codice dispositivo fisico CENTRALINO
 - Codice per microcontrollore Arduino.
 - Breve spiegazione.
 - Problematiche principali.
 - Codice applicazione CENTRALINO
 - Codice descrittivo della grafica dell'applicazione.
 - Codice operativo dell'applicazione.

- Breve spiegazione.
 - Problematiche principali.
- Conclusioni :
 - Futuri sviluppi.
 - Dedic.
 - Fonti.

Dispositivo fisico CENTRALINO

Documentazione generale di progetto :

- Idea del progetto.
 - Il progetto nasce con l'intenzione di rendere qualsiasi plesso abitativo, residenziale o lavorativo più connesso con le persone stesse che lo popolano. Il progetto è il tentativo di portare ad un nuovo il livello di controllo dell'ambiente per l'uomo. CENTRALINO rende a portata di mano il controllo di tutto ciò che di elettrico ci circonda. È possibile controllare elettrodomestici, strumenti di lavoro, strumenti musicali, fornelli elettrici, impianti di aereazione, condizionatori, sistemi di riscaldamento e così via tutto ciò che necessita di elettricità per funzionare. Questa centralizzazione di controllo può in alcuni casi cambiare totalmente l'esperienza di una persona con il mondo circostante, per esempio nel caso di una persona disabile. La comodità dell'automatizzazione a volte non è solo una comodità è bensì una necessità. In questi anni il tema della cosa domotica è un tema in cui le aziende si stanno confrontando duramente per offrire sempre una maggiore automazione e integrazione con sistemi domotici. La domotica è un complesso di moltissime tecnologie diverse e richiedono una forte interdisciplinarietà si parla di ingegneria edile, architettura, ingegneria energetica, automazione, elettrotecnica, elettronica, telecomunicazioni, informatica e design. Il progetto CENTRALINO come la domotica in generale ambisce perciò a:
 - Migliorare la qualità della vita.
 - Migliorare la sicurezza.

- Semplificare la progettazione, l'installazione, la manutenzione e l'utilizzo della tecnologia.
 - Ridurre i costi di gestione.
 - Convertire i vecchi ambienti e i vecchi impianti in ambienti più dinamici.
- Caratteristiche.
 - Il nome del progetto è "CENTRALINO", prende il nome proprio dal tipo di operazione che fa, sia "centralizzare" il controllo ma anche gestirlo, come accade in telefonia. Consiste in un dispositivo che permette il controllo remoto di un presa di più prese di corrente con la possibilità di essere programmabile ad intervalli di tempo grazie ad un orologio interno. Il dispositivo si configura facilmente con i dispositivi mobili come telefoni, tablet o computer. Quest'ultimo deve solamente supportare applicazioni create per sistemi operativi Android ed essere provvisto di trasmettitore Bluetooth.
 - Settore di riferimento : nuove tecnologie, Internet of Things (Internet delle Cose), domotica.
 - Funzioni
 - Le funzioni per ora sono 3 ma la cosa più importante è che ormai c'è una base che chiunque può usare per sviluppare le proprie funzioni personalizzate. Le funzioni sono :
 - 1) Spegnimento/Accensione istantaneo/a – Per una maggiore sicurezza in futuro saranno implementate misure di sicurezza secondo le quali non sarà possibile accendere e spegnere in continuazione la stessa prese per evitare

di danneggiare la componentistica del dispositivo collegato e di CENTRALINO stesso.

- 2) Spegnimento/Accensione con sveglia. Sfrutta l' RTC per confrontare il tempo di CENTRALINO con l'orario che viene impostato dall'utente che imposta la "sveglia".
 - 3) Spegnimento/Accensione con timer. Sfrutta la libreria <Time.h> che permette l'utilizzo di un oggetto (strumento informatico) di immagazzinamento di orario, data e fondamentale permette di avere un orario locale sempre aggiornato anche senza un dispositivo come l'RTC. Interrogare in continuazione l'RTC significherebbe aumentarne troppo il consumo di batteria (la batteria dell'RTC) e siccome richiede tempo cambiarla si preferisce usare la libreria citata e aumentare quindi la durata della batteria. Il dispositivo fisico CENTRALINO riceve dall'applicazioni software CENTRALINO in minuti la durata del timer e esegue delle operazione per sapere a quante ore, minuti e secondi corrispondono.
- Disegno forma e cianografia.

- Interfaccia utente.
 - L'interfaccia utente di CENTRALINO è prettamente manuale, è possibile solo la disattivazione completa di esso tramite il pulsante blu all'interno della scatola contenitrice. CENTRALINO è programmabile ed è possibile interagire con esso solo con l'applicazione software appositamente a meno che non lo si voglia spegnere tramite il suddetto pulsante interno.
- Elenco componenti.

Nome	Venditore	Prezzo (€)	Breve descrizione
Modulo relay da 2 canali	Sain Smart	9.00	Modulo per il controllo del flusso di corrente.
Arduino Micro	Ali-express	4.50	Microcontrollore
Modulo Bluetooth HC-06	Ali-express	5.20	Modulo per la trasmissione senza fili Bluetooth
Modulo RTC (Real Time Clock)	Ali-express	3.50	Modulo orologio indipendente e autonomo (alimentato a batteria)
Scatola	Tomea plastiche	7.00	Scatola contenitrice.

Tabella 1

- Scelta dei componenti e preventivo.
 - I componenti hanno uno scopo prototipale, non si esclude affatto la futura rielaborazione e nuova creazione di una lista di componenti più specifici.
 - La scatola contenitrice è appositamente stata realizzata in plexiglass perché è il materiale che più si avvicina alla necessità di un materiale esteticamente accattivante, parzialmente ignifugo e relativamente facilmente realizzabile. Sono stati esclusi legno e metallo, il primo per la sua predisposizione al fuoco e il secondo per la sua nota conduttività elettrica.

- Il preventivo per la realizzazione dello stesso progetto è di 29.2 € per quanto riguarda la componentistica che è stata utilizzata. Supponendo una produzione in serie il prezzo calerebbe drasticamente. Una delle forze infatti del progetto stesso è anche il basso costo di costruzione che esso richiede.
- Simulazioni.
 - Sono state effettuate delle simulazioni in ambienti controllati e sicuri con esiti positivi, testando tutte le funzionalità di CENTRALINO.

Scheda di progetto

- Analisi dei requisiti.
 - CENTRALINO è stato pensato per funzionare con i mezzi di comunicazione senza fili. Requisito basilare è perciò un mezzo di comunicazione senza fili, la versione attuale sfrutta la tecnologia Bluetooth ma lascia aperte le possibilità all'utilizzo di altre non essendo strettamente dipendente da essa.
 - CENTRALINO è stato pensato per funzionare come una rete di dispositivi che collaborano nello stesso ambiente. Perciò il protocollo di messaggistica tra i vari CENTRALINO si è sviluppato intorno a questa necessità, provvedendo a far sì che fosse possibile avere molti dispositivi tutti collegati contemporaneamente. Come avviene lo scambio di pacchetti di dati tra i vari CENTRALINO verrà spiegato successivamente nell'apposito paragrafo.
- Analisi delle competenze necessarie.
 - Inglese:
 - Le ricerche necessarie a comprendere e documentarsi sulle strumentazioni più adatte a dar vita a questo progetto sono in vasta preponderanza presenti sul web in lingua inglese. Senza la conoscenza della lingua fare ricerche specifiche sarebbe quasi impossibile. C'è inoltre grande bisogno di una conoscenza di termini tecnici della lingua poiché si utilizza un determinato tipo di lessico nelle documentazioni al proposito di apparecchi elettronici.
 - Fisica generale ed elettronica:

- Conoscenza necessaria per un sicuro lavoro manuale con correnti ad alte tensioni come quelle con cui CENTRALINO lavora. Conoscenza del funzionamento di un circuito e dei possibili corretti posizionamenti dei relay magnetici. In più sono necessarie le conoscenze di sintassi informatica e logica specifiche dei circuiti, indispensabili per il funzionamento dei programmi.
- Informatica:
 - Vengono utilizzati per la produzione del progetto i seguenti linguaggi di programmazione :
 - C.
 - C++.
 - Conoscenze base di java.
 - Essi non sono strettamente collegati allo studio effettuato a scuola, tuttavia i metodi per interfacciarsi ed imparare nuovi linguaggi sono in maggior parte stati appresi con lo studio in aula.
- Elettronica pratica:
 - Conoscenze tecniche per l'assemblaggio, la saldatura dei componenti e la realizzazione del circuito.
- Italiano:
 - Capacità descrittive per la più adeguata forma di descrizione e spiegazione per un progetto che necessita di alte capacità lessicali per non fraintenderne il funzionamento. Necessarie per la compilazione di un manuale tecnico facilmente interpretabile.

- Tecnologia progettazione di sistemi:
 - Conoscenze di tecniche di base e approfondite per lo sviluppo di:
 - Software multi-piattaforma.
 - Conoscenza dei sistemi di trasferimento dati necessari al trasferimento senza fili (Bluetooth ora ma futuramente verrà sviluppato anche WI-FI, e onde-radio a meno di 2.5 Ghz) protocollato a pacchetti.
 - Conoscenza del sistema ISO/OSI.
 - Interfaccia per l'utente.
 - Interfacciamento tra il sistema operativo del dispositivo fisico CENTRALINO e il software applicativo CENTRALINO (sul dispositivo Android).
- Le sottocategorie sono innumerevoli perciò queste due grandi categorie a simbolo delle vaste conoscenze necessarie.
 - Organizzazione d'impresa:
 - Possibili e probabili risvolti futuri non ancora presi in considerazione per mancanza di tempo.
 - Capacità manuali e di inventiva:
 - Da non sottovalutare.

Architettura di sistema

- Un utente può collegarsi a CENTRALINO con l'apposita applicazione software semplicemente tramite le impostazioni Bluetooth offerte dall'applicazione stessa. Una volta che l'utente si connette ad un CENTRALINO, per controllarne un altro dovrà prima

disconnettersi da quello a cui è già collegato. Il sistema finale si baserà sulla presenza di tre personaggi :

- Utente controllore (utilizza applicazione software descritta nella sezione successiva della relazione) che impartisce i comandi ai dispositivi CENTRALINO.
 - CENTRALINO (descritta in questa parte della relazione) che controlla la presa di corrente.
 - CENTRALINO-SERVER (non descritta in questa relazione ma di cui è prevista la futura creazione) che dà la possibilità all'utente di controllare tutti i CENTRALINO associati senza mai doversi disconnettere da CENTRALINO SERVER che provvederà lui stesso a gestire le comunicazioni con gli altri.
- Specifiche implementative
 - Per funzionare ed essere implementato in un sistema più grande, CENTRALINO necessita solamente di una connessione alla rete elettrica alimentata a 220 V (Volt) (come ad esempio la rete elettrica domestica) ed una tensione di corrente non superiore a 15 A (Ampere) per non rischiare di danneggiare il circuito o incorrere in un malfunzionamento del dispositivo.

Documento tecnico :

- Documentare il completamento del progetto e tabella sui tempi delle attività.
 - CENTRALINO si trova al momento nella sua versione ultima ma come già espresso in questo documento sono già previste future migliorie. Il progetto è stato pensato a Febbraio 2017 e la sua realizzazione prototipale è giunta al termine in Giugno 2017 . Le macro fasi del completamento del progetto sono state:

Fase	Durata
<ul style="list-style-type: none">• Ideazione del progetto e stesura di un documento iniziale dove si chiariscono gli obiettivi e le funzionalità del progetto che si intende realizzare.	2 settimane ca.
<ul style="list-style-type: none">• Analisi dei requisiti.	1 settimana ca.
<ul style="list-style-type: none">• Analisi delle peculiarità tecniche dei requisiti.	1 settimana ca.
<ul style="list-style-type: none">• Ricerche di natura tecnica sulle caratteristiche dei componenti e le loro funzionalità, in particolare moduli e ricerche sul funzionamento delle reti elettriche.	2 settimane ca.
<ul style="list-style-type: none">• Scelta dei componenti.	3 giorni ca.
<ul style="list-style-type: none">• Ideazione e progettazione del software residente nel microcontrollore che gestisce tutti i componenti collegati.	1 settimana ca.
<ul style="list-style-type: none">• Lunga e dolorosa creazione del software (trattata nell'apposito paragrafo).	1 mese ca.
<ul style="list-style-type: none">• Assemblaggio dei componenti.	1 settimana ca.
<ul style="list-style-type: none">• Test e cura dei difetti di creazione, hardware e software, ma soprattutto software.	1 settimana ca.
<ul style="list-style-type: none">• Ideazione dell'applicazione software utile al controllo di CENTRALINO	4 giorni ca.
<ul style="list-style-type: none">• Progettazione e realizzazione del sistema di scambio dati tra : applicazione e CENTRALINO; tra CENTRALINO e CENTRALINO SERVER..	2 settimane ca.
<ul style="list-style-type: none">• Scelta dei criteri per la creazione dell'apparato grafico e successiva creazione.	1 settimana ca.
<ul style="list-style-type: none">• Realizzazione del sistema software utilizzato dall'applicazione.	1 settimana ca.

• Test di funzionamento dell'applicazione e correzione di errori grafici e software	1 settimana ca.
• Test di utilizzo dell'applicazione all'opera in casi reali d'uso (connessa a CENTRALINO).	1 settimana ca.
• Progettazione della scatola contenitrice di CENTRALINO.	2 giorni ca.
• Creazione di questo documento	2 giorni ca.

Tabella 2

- Strumenti utilizzati (Software).
 - Fritzing: è un software libero per la progettazione elettronica, focalizzato sul passaggio da semplici prototipi al circuito stampato da inviare alla produzione. Utile a disegnare il prototipo del circuito ancora prima che venga costruito.
 - Arduino IDE: è l'ambiente di sviluppo integrato (Integrated Development Enviroment) di Arduino, è un'applicazione multiplatforma in Java usata per compilare il codice con cui viene programmato il microcontrollore che gestisce i sensori e gli altri componenti.
 - AliExpress: è un ramo dell'azienda "Alibaba Group" che unisce imprese cinesi e offre un servizio di vendita al dettaglio a un mercato internazionale, rivolgendosi prevalentemente a compratori europei. E' il fornitore dei componenti.
 - MIT App Inventor (Framework): è una applicazione web creata da Google ma ora posseduto dal Massachusetts Institute of Technology. Un ambiente di sviluppo utile a creare applicazioni per android di uso personale.
 - BlueStacks : è un'applicazione creata dall'azienda americana BlueStacks Systems Inc. nel 2011. BlueStacks App Player è un emulatore di app android per i sistemi operativi MacOS e Windows. Le caratteristiche principali sono l'ambiente completamente personalizzabile, il supporto per molteplici configurazioni di sistema operativo e l'integrazione con Google Play.

- Strumenti utilizzati (Hardware):
 - Saldatore.
 - Cacciaviti vari.
 - Stagno.
 - Silicone liquido.
 - Strumenti di misura di precisione.
- Spiegazione del codice e le sue funzionalità e particolarità.
 - (Si trova nella sezione apposita "codici e relative spiegazioni").
- Problematiche degne di nota affrontate nella creazione del codice.
 - (Si trova nella sezione apposita "codici e relative spiegazioni").
- Manutenzione.
 - CENTRALINO è completamente autonomo e non necessita di manutenzione. Se si desidera effettuare un aggiornamento del software è possibile farlo connettendo il microcontrollore di CENTRALINO ad un qualsiasi dispositivo provvisto di compilatore ARDUINO IDE e scaricando il codice open source fornito. Il codice aggiornato di CENTRALINO si trova sul sito di hosting GitHub all'indirizzo web:
 - " <https://github.com/CENTRALINOproject> "

Applicazione software CENTRALINO

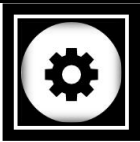
Documentazione generale di progetto :

- Idea del progetto.
 - L'idea di CENTRALINO si concretizza solamente con la creazione di un mezzo che ne rende immediato l'utilizzo. La soluzione per questo progetto è stata quella di creare un'applicazione che rendesse intuitivo l'utilizzo del dispositivo e di conseguenza l'utilizzo delle sue funzionalità. Nulla esclude che un domani CENTRALINO non possa essere controllato da un qualsiasi dispositivo capace di trasmettere messaggi Bluetooth.
- Caratteristiche.
 - L'applicazione riconosce da se stessa se il dispositivo su cui è eseguita può supportare le sue funzionalità e le quando non lo è lo segnala con notifiche all'utente.
 - Per essere il più intuitiva possibile l'applicazione è stata creata tenendo in mente un principale obbiettivo fisso ovvero "la semplicità". Si desiderava che l'utente trovandosi davanti ad una nuova interfaccia non si trovasse mai spiazzato e perciò è stata adottata una tecnica stilistica sviluppatasi fin dalla metà del 1900, ovvero il minimalismo (Minimal art). Le icone risultano perciò quasi scarse e i colori sono ridotti al minimo (bianco e nero).
- Interfaccia utente.
 - Nelle seguenti tabelle e paragrafi si mostrano le funzionalità del "Menù a scorrimento" e del "pannello dei pulsanti". Nelle seguenti tabelle si parla spesso di "accendere" / "spegnere" / "accensione" / "spegnimento"; in questi casi ci si

riferisce sempre all'accensione e lo spegnimento dei dispositivi connessi al dispositivo fisico CENTRALINO.

- L'interfaccia deve dare modo all'utente di impostare le tre attività offerte dall'applicazione, ovvero : Spegnimento/accensione istantaneo/a, Sveglia (permette di impostare l' accensione/spegnimento ad orari precisi), Timer (permette di impostare l' accensione/spegnimento ad un intervallo di tempo preciso).
- L'interfaccia è stata completamente realizzata partendo da zero. Persino le icone sono state curate poiché era impossibile trovare immagini che fossero "adatte" o che in generale calzassero per l'effetto che si intendeva ottenere.
- L'interfaccia utente si struttura di 2 schermate principali (il codice si trova nel paragrafo successivo):
 - Home (pagina iniziale).
 - Schermata delle attività (nel codice "Default_screen").
- La "schermata delle attività" si può a sua volta scomporre in due macro aree:
 - Menù a scorrimento (nel codice "Slider_container").
 - Pannello dei pulsanti (nel codice "Button_panel"). Permette di accedere alle varie aree del "Menù a scorrimento".

Menù a scorrimento

Nome	Icona (se presente)	Nome nel codice	Funzione
Imposta manualmente		Sveglia / Timerpicker_timer	Permette di scegliere manualmente l'orario a cui si vuole impostare l'accensione o lo spegnimento .








Interruttore acceso/spento		Toggle_on_off	Fa sì che all'orario impostato dell'utente CENTRALINO spenga o accenda.
Pulsante di conferma		Set_clock_send / Confirm_timer	Conferma l'attività che si vuole e la imposta su CENTRALINO.
Slider del tempo		Slider_canvas	Permette di impostare manualmente il tempo del timer.
Pulsante spegnimento/accensione istantaneo/a		Hexagon_button	Permette di spegnere/accendere Istantaneamente. Il colore del pulsante cambia in base allo stato della presa di corrente.
Pulsante comandi vocali		Hexagon_button	Permette di controllare con la voce i dispositivi collegati a CENTRALINO utilizzando i comandi "spegni (nome dispositivo)" e "accendi (nome dispositivo)".
Pulsante di sincronizzazione		Sync_button	Se premuto effettua la sincronizzazione dei dispositivi connessi al dispositivo fisico CENTRALINO con l'applicazione.

Tabella 3

Pannello dei pulsanti

Nome	Icona (se presente)	Nome nel codice	Funzione
Apri-sezione destra		Right_button	Apri la sezione destra del menù, la sezione per la funzione Timer.



Selezione dispositivo		Bluetooth_picker	Apre il menù a lista che permette di scegliere tra i dispositivi associati tramite Bluetooth.
Apri-sezione sinistra		Left_button	Apri la sezione destra del menù, la sezione per la funzione Sveglia.

Tabella 4

- Simulazioni.
 - Sono state effettuate delle simulazioni sull'emulatore di sistemi Android BlueStacks versione 2.0.8.5638 e sul dispositivo Samsung S5 con sistema operativo Android versione 5.0 con esiti positivi.
- Architettura di sistema
 - Si rimanda alla lettura del paragrafo sull'architettura di sistema presente nel capitolo "Dispositivo fisico CENTRALINO".
- Specifiche implementative
 - Per funzionare ed essere implementato l'applicazione CENTRALINO necessita solamente di una connessione Bluetooth con il dispositivo fisico CENTRALINO e che il dispositivo su cui è installata l'applicazione supporti l'applicazione stessa (sistemi operativi Android 4.0 / 5.0 / 6.0).

Documento tecnico :

- Si rimanda alla lettura della Tabella 2 presente nel capitolo "dispositivo fisico CENTRALINO".
- Codice (Si trova nella sezione apposita "codici e relative spiegazioni").

- Spiegazione del codice e le sue funzionalità e particolarità (Si trova nella sezione apposita "codici e relative spiegazioni").
- Problematiche degne di nota affrontate nella creazione del codice (Si trova nella sezione apposita "codici e relative spiegazioni").
- Manutenzione.
 - L'applicazione CENTRALINO non necessita di manutenzione e verrà aggiornata ogni volta che si troverà un difetto o verrà introdotta una nuova funzione. Essa potrà essere scaricata gratuitamente dalla pagina di GitHub ufficiale di CENTRALINO citata nel seguente punto.
 - L'applicazione di CENTRALINO si trova al seguente indirizzo web e per scaricarla non è necessaria nessuna registrazione.
 - " <https://github.com/CENTRALINOproject> "

Licenze.

- Il software di CENTRALINO è registrato secondo la MIT License. La MIT License "è una licenza permissiva, cioè permette il riutilizzo nel software proprietario sotto la condizione che la licenza sia distribuita con tale software."



- Il software dell'applicazione di CENTRALINO è anche esso distribuito secondo la stessa licenza. In più bisogna specificare che l'applicazione è stata creata con l'applicazione web "MIT AppInventor 2.0 Beta" che si basa sulle teorie dell'apprendimento del costruzionismo ("Il costruzionismo è basato sulla teoria del costruttivismo secondo la quale l'individuo che apprende costruisce modelli mentali per comprendere il mondo intorno a lui. Il costruzionismo sostiene che l'apprendimento avviene in modo più efficiente se chi apprende è coinvolto nella produzione di oggetti tangibili"). MIT AppInventor gode della sua propria licenza, le applicazioni create con esso sono perciò soggette anche ad essa.
- Per quanto riguarda la forma fisica di CENTRALINO, la componentistica di CENTRALINO sono aspetti non ancora tutelati da nessuna licenza o brevetto data la non totale certezza di quali componenti si utilizzeranno in futuro, tuttavia quando verranno rese di pubblico dominio le caratteristiche saranno distribuite secondo la licenza [Attribution-ShareAlike 4.0 International \(CC BY-SA 4.0\)](#)



Codici e relative spiegazioni.

Codice dispositivo fisico CENTRALINO

- Codice per microcontrollore Arduino.

{Versione del codice CLIENTinoBETA1.4.ino}

```
#include <MemoryFree.h>
#include <SoftwareSerial.h>
#include <IRremote.h>
#include <string.h>
#define rxPin 9
#define txPin 10
#define RECV_PIN 4

/*****PLEASE MODIFY THIS LINE WHEN NEEDED*****/
#define total_relay 2 // Total number of relay connected

#include <Wire.h>
#include "RTCLib.h"
#include <TimeAlarms.h>
#include <TimeLib.h>
#include <Time.h>

/*****RTC*****/
RTC_DS1307 rtc;
tmElements_t tm;

#if defined(ARDUINO_ARCH_SAMD)
// for Zero, output on USB Serial console, remove line below if using programming port to
program the Zero!
#define Serial SerialUSB
#else
#define Serial Serial
#endif

/*****BLUETOOTH*****/
SoftwareSerial bluetooth(rxPin, txPin);

/*****Infra-Red*****/
IRrecv irrecv(RECV_PIN);
decode_results results;

String bluetooth_Buffer;
```

```

struct function
{
int activity;
int extra_value;
int extra_value2;
} function_temp;

struct relay
{
int socket = 0;
int hour_daily_end = 61;
int minute_daily_end = 61;
int second_daily_end_and_start = 0;
int hour_daily_start = 61;
int minute_daily_start = 61;

int timer_off_hour = 61;
int timer_off_minute = 61;
int timer_off_second = 61;

int timer_on_hour = 61;
int timer_on_minute = 61;
int timer_on_second = 61;

int description = 0;
} rel[total_relay];

void all_rel_HIGH()
{
for (int i=0; i<total_relay ;i++)
{
digitalWrite(rel[i].socket, HIGH);
Serial.print("Impostato ad HIGH pin n. ");
Serial.print(rel[i].socket);
Serial.print(" per struttura rel[] n. ");
Serial.println(i);
}
}

void init_output_rel()
{

```

```

for (int i=0; i<total_relay ;i++)
{
pinMode(rel[i].socket, OUTPUT);
Serial.print("Inizializzato pin n. ");
Serial.print(rel[i].socket);
Serial.print(" per struttura rel[] n. ");
Serial.println(i);
}
}

void setup ()
{
/*
#ifdef ESP8266
while (!Serial); // for Leonardo/Micro/Zero
#endif
*/
Serial.begin(9600); //set baud rate
bluetooth.begin(9600);
init_output_rel(); //Declare output all relays
all_rel_HIGH(); //Set HIGH all relays
delay(1000);

irrecv.enableIRIn(); // Start the receiver
//FOR DEBUG
/* Serial.begin(57600);
if (! rtc.begin()) {
Serial.println("Couldn't find RTC");
while (1);
}
*/

if (! rtc.begin()) {
Serial.println("Couldn't find RTC");
while (1);
}
if (! rtc.isrunning()) {
Serial.println("RTC is NOT running!");
rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
// rtc.adjust(DateTime(year, month, day, hour, min, sec)); IF NEED ADJUSTMENTS
}
DateTime now = rtc.now();
setTime(now.hour(),now.minute(),now.second(),now.month(),now.day(),now.year());

/*****PLEASE HERE DECLARE THE PIN'S OF THE RELAYS, ONE EACH REL[] STRUCTURE*****/
//example : rel[i].socket = pin connected to relay ;

```

```

rel[0].socket = 6;
rel[0].description = 1;
rel[1].socket = 7;
rel[1].description = 3;

}

void printDigits(int digits)
{
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

void digitalClockDisplay()
{
  // digital clock display of the time
  DateTime now = rtc.now();
  Serial.print(now.hour());
  printDigits(now.minute());
  printDigits(now.second());
  Serial.println();
}

bool ir_handler(decode_results *results)
{
  {
    if (results->value == 0xA90 )
      turn_off();
    else
      return false;
  }
}

void bluetooth_handler()
{
  {
    char check = ' ';
    check = bluetooth.read();
    if ((int(check) != 13) && (int(check) != 10) && (int(check) != 32)) //Ignore '\r' , '\n' and spaces
      bluetooth_Buffer += check;
    else
    {
      Serial.println("Relivated unprintable chars, ASCII value is : ");
      Serial.println(int(check));
    }
    Serial.print("Value of bluetooth buffer is = ");
    Serial.print(bluetooth_Buffer);
    Serial.println("");
  }
}

```

```

if (check==''])
bluetooth_Parser();
}

```

```

/*****MEANINGS OF DESCRIPTIONS*****/
1 = Lamp
2 = Chandelier
3 = Computer
4 = Fridge
5 = Fan
6 = Microwave
7 = Charger
8 = Electric shutter
9 = Amplifier
10 = Speakers
11 = Aircooler
12 = Confetti's cannon
*****/

```

```

/*****STRUCTURE BLUETOOTH PACKAGE*****/
* ('figures' in italian : 'cifre')
* ([]) Message opening 1 position : [0]
* (1st number, two figures) Activity 2 position : [2]
* (,) 1 position : [3]
* (2nd number , four figures) Extra_value 4 position : [7]
* (,) 1 position : [8]
* (3rd number , six figures) Extra_value2 6 position : [14]
* (]) message closer 1 position : [15] Total = 16
*****/

```

```

void bluetooth_Parser()
{ Serial.println("Open bluetooth parser");
if (int(bluetooth_Buffer.length())!=16) //Hard coded dimensions and particularities of the message
{Serial.println("Lenght check not passed");
Serial.println(int(bluetooth_Buffer.length()));
bluetooth_Buffer = "";}
else if (bluetooth_Buffer[0]!='[')
{Serial.println("Opener check not passed");
Serial.println(bluetooth_Buffer[0]);
bluetooth_Buffer = "";}
else if (bluetooth_Buffer[3]!='(',')')
{Serial.println("First comma check not passed");
Serial.println(bluetooth_Buffer[3]);
bluetooth_Buffer = "";}
else if (bluetooth_Buffer[8]!='(',')')
{Serial.println("Second comma check not passed");
Serial.println(bluetooth_Buffer[8]);
bluetooth_Buffer = "";}
}

```



```

else if (bluetooth_Buffer[15]!='\'])
{Serial.println("Closer check not passed");
Serial.println(bluetooth_Buffer[15]);
bluetooth_Buffer = "";}
else
{ String temp_string;
long temp=0;

Serial.print("Received package : ");
Serial.println(bluetooth_Buffer);
temp_string=bluetooth_Buffer.substring(1,3); // substring (from [index] ,to [index we want in the sub +1])// https://www.arduino.cc/en/Reference/StringSubstring
temp=atoi(temp_string.c_str());
function_temp.activity = int(temp);
Serial.print("Received activity : ");
Serial.println(function_temp.activity);
temp_string=bluetooth_Buffer.substring(4,8);
temp=atoi(temp_string.c_str());
function_temp.extra_value = int(temp);
Serial.print("Received extra_val : ");
Serial.println(function_temp.extra_value);
temp_string=bluetooth_Buffer.substring(9,15);
temp=atoi(temp_string.c_str());
function_temp.extra_value2 = int(temp);
Serial.print("Received extra_val2 : ");
Serial.println(function_temp.extra_value2);

bluetooth_Buffer=""; //Reset buffer for new packets

activity_manager();
}
}

// ACTIVITY MANAGER
void activity_manager ()
{ Serial.println("Open activity manager");
Serial.println(function_temp.activity);
digitalClockDisplay();
if (function_temp.activity == 1) // Turn OFF relay n. extra_value
turn_off();
if (function_temp.activity == 2 ) // Turn On relay n. extra_value
turn_on();

if (function_temp.activity == 3 ) // Set timer to turn ON relay n. extra_value in time extra_value2

```

```

set_timer_on();

if (function_temp.activity == 4 ) // Set timer to turn OFF relay n. extra_value in time
extra_value2
set_timer_off();
if (function_temp.activity == 5 ) // Set daily START on relay n. extra_value in time
extra_value2
set_daily_start();

if (function_temp.activity == 6 ) // Set daily END on relay n. extra_value in time
extra_value2
set_daily_end();

if (function_temp.activity == 50 ) // REQUEST of OFFER
bluetooth_offer();

// 51 is an OFFER package (Will never be received on CLIENTino)
// 99 is a CLOSE communication_packet (Will never be received on CLIENTino)*//

return;
}

void bluetooth_offer()
{
Serial.println("Offer now :");
for (int i=0; i<total_relay; i++)
send_package(51, rel[i].socket, rel[i].description );
send_package(99, 0, 0); //Close communication
return;
}

void send_package(int activity, int extra_value, int extra_value2)
{ //Serial.println("FLAG send package");
//Serial.println("Send package");
//Serial.print("freeMemory()=");
//Serial.println(freeMemory());
if ((activity>99)|| (extra_value>9999)|| (extra_value2>999999))
{
Serial.println("Unable to send the packet because of uncorrect values");
}
else
{

```

```

String packet = "";
String aux_activity = zero_padd(activity, 2);
//Serial.println("THIS IS AUX ACT");
//Serial.println(aux_activity);
String aux_extra_value = zero_padd(extra_value, 4);
//Serial.println("THIS IS AUX EXTRA");
//Serial.println(aux_extra_value);
String aux_extra_value2 = zero_padd(extra_value2, 6);
//Serial.println("THIS IS AUX EXTRA 2");
//Serial.println(aux_extra_value2);

```

```

packet += '[';
packet += aux_activity;
packet += ',';
packet += aux_extra_value;
packet += ',';
packet += aux_extra_value2;
packet += ']';

```

```

Serial.print("This is packet");
Serial.println(packet); //packet.concat(aux_activity)
bluetooth.print(packet);
}
return;
}

```

```

String zero_padd(int value, int spaces)
{ Serial.println("FLAG zero_padd");
String result="";
Serial.println("FLAG zero_padd1");
int padds=spaces, aux_value=value;
Serial.println("FLAG zero_padd2");

```

```

if(aux_value!=0)
{
do
{
Serial.println("FLAG zero_padd3");
Serial.println(aux_value);
if (aux_value > 0)
{
aux_value = aux_value / 10;
Serial.println("This is aux after /10");
Serial.println(aux_value);
padds--;
}
}
}

```

```

}
while (aux_value > 0);

}
else
{
padds=spaces-1;
}
Serial.print("This is value :");
Serial.println(value);
Serial.print("This is spaces :");
Serial.println(spaces);
Serial.print("This is padds :");
Serial.println(padds);
if (padds!=0)
{
if (padds==1)
result="0";
else if (padds==2)
result="00";
else if (padds==3)
result="000";
else if (padds==4)
result="0000";
else if (padds==5)
result="00000";
else if (padds==6)
result="000000";
}

result.concat(value);

return result;
}

void set_timer_off()
{
rel[function_temp.extra_value].timer_off_hour = hour()+function_temp.extra_value2/3600;
rel[function_temp.extra_value].timer_off_minute = minute()+function_temp.extra_value2/60;
rel[function_temp.extra_value].timer_off_second = second()+function_temp.extra_value2%60;
adjust_timer_off();
}

void set_timer_on()
{

```

```

rel[function_temp.extra_value].timer_on_hour = hour(); //+function_temp.extra_value2; ///3600;
rel[function_temp.extra_value].timer_on_minute = minute(); //+function_temp.extra_value2; ///60;
rel[function_temp.extra_value].timer_on_second = second()+function_temp.extra_value2 ; ///60;
adjust_timer_on();
}

```

```

void adjust_timer_on()
{
if(rel[function_temp.extra_value].timer_on_second >= 60)
{
rel[function_temp.extra_value].timer_on_minute =
rel[function_temp.extra_value].timer_on_minute +
rel[function_temp.extra_value].timer_on_second / 60;
rel[function_temp.extra_value].timer_on_second =
rel[function_temp.extra_value].timer_on_second % 60;
}
}

```

```

if(rel[function_temp.extra_value].timer_on_minute >= 60)
{ rel[function_temp.extra_value].timer_on_hour = rel[function_temp.extra_value].timer_on_hour
+ rel[function_temp.extra_value].timer_on_minute / 60 ;
rel[function_temp.extra_value].timer_on_minute =
rel[function_temp.extra_value].timer_on_minute % 60;
}
}

```

```

if(rel[function_temp.extra_value].timer_on_hour >= 24)
rel[function_temp.extra_value].timer_on_hour = rel[function_temp.extra_value].timer_on_hour %
24;

```

```

}

```

```

void adjust_timer_off()
{
if(rel[function_temp.extra_value].timer_off_second >= 60)
{
rel[function_temp.extra_value].timer_off_minute =
rel[function_temp.extra_value].timer_off_minute +
rel[function_temp.extra_value].timer_off_second / 60;
rel[function_temp.extra_value].timer_off_second =
rel[function_temp.extra_value].timer_off_second % 60;
}
}

```

```

if(rel[function_temp.extra_value].timer_off_minute >= 60)

```

```

{ rel[function_temp.extra_value].timer_off_hour =
rel[function_temp.extra_value].timer_off_hour +
rel[function_temp.extra_value].timer_off_minute / 60 ;
rel[function_temp.extra_value].timer_off_minute =
rel[function_temp.extra_value].timer_off_minute % 60;
if(rel[function_temp.extra_value].timer_off_hour >= 24)
rel[function_temp.extra_value].timer_off_hour =
rel[function_temp.extra_value].timer_off_hour % 24;
}
}

void set_daily_end()
{
rel[function_temp.extra_value].hour_daily_end = function_temp.extra_value2/100;
rel[function_temp.extra_value].minute_daily_end = function_temp.extra_value2%100;
}

void set_daily_start()
{
rel[function_temp.extra_value].hour_daily_start = function_temp.extra_value2/100;
rel[function_temp.extra_value].minute_daily_start = function_temp.extra_value2%100;
}

int match_time(int i)
{ int result=0; //No match founded

int hour_now = hour();
int minute_now = minute();
int second_now = second();
if ( (rel[i].hour_daily_end==hour_now) && (rel[i].minute_daily_end==minute_now) ) //return 6
(same n. as activity)
if ( rel[i].second_daily_end_and_start==second_now)
{

Serial.println(" match with daily_end, I'll return 6");

result=6;
}
if ( (rel[i].hour_daily_start==hour_now) && (rel[i].minute_daily_start==minute_now)) //return
5 (same n. as activity)
if( rel[i].second_daily_end_and_start==second_now )
{
Serial.println(" match with daily_start, I'll return 5");
result=5; //Turn on founded

```

```

}
if ( (rel[i].timer_off_hour==hour_now) && (rel[i].timer_off_minute==minute_now ) ) //return 4
(same n. as activity)
if (rel[i].timer_off_second==second_now)
{
Serial.println(" match with timer_off, I'll return 4");
rel[i].timer_off_hour = 61;
rel[i].timer_off_minute = 61;
rel[i].timer_off_second = 61;
result=4;
}
if ( (rel[i].timer_on_hour==hour_now ) && (rel[i].timer_on_minute==minute_now) ) //return 3
(same n. as activity)
if( rel[i].timer_on_second==second_now )
{
Serial.println(" match with timer_on, I'll return 3");
rel[i].timer_on_hour = 61;
rel[i].timer_on_minute = 61;
rel[i].timer_on_second = 61;
result=3;
}

return result;
}

```

```

void turn_by_match()
{ Serial.println("Checking by match : ");
for (int i=0; i<total_relay; i++)
{int match_result = match_time(i);
if (match_result == 6) //daily_end
turn_off_pin(i);
else if (match_result == 5) //daily_start
turn_on_pin(i);
else if (match_result == 3) //timer_on
turn_on_pin(i);
else if (match_result == 4) //timer_off
turn_off_pin(i);
else Serial.println("No match founded");
}
}

```

```

void turn_on_pin(int index)
{
digitalWrite( index , LOW);
Serial.println("ON");
}

```

```

delay (4000);
}

void turn_off_pin(int index)
{
digitalWrite( index , HIGH);
Serial.println("OFF");
delay (4000);
}

void turn_off()
{
digitalWrite(function_temp.extra_value , HIGH);
Serial.println("OFF");
delay (4000); //this is to not overcharge the relay
}

void timer_turn_off()
{
digitalWrite( rel[function_temp.extra_value2].socket , HIGH);
Serial.println("OFF");
delay (4000); //this is to not overcharge the relay
}

void timer_turn_on()
{
digitalWrite( rel[function_temp.extra_value2].socket , LOW);
Serial.println("ON");
delay (4000); //this is to not overcharge the relay
}

void turn_on()
{
digitalWrite(function_temp.extra_value , LOW);
Serial.println("ON");
delay (4000); //this is to not overcharge the relay
}

void digitalClockDisplay_notRTC()
{
// digital clock display of the time

```



```

Serial.print(hour());
printDigits(minute());
printDigits(second());
Serial.println();
}

void loop ()
{
if(blueetooth.available())
bluetooth_handler();
if (irrecv.decode(&results))
ir_handler(&results);

digitalClockDisplay_notRTC();
turn_by_match();
Serial.print("freeMemory()=");
Serial.println(freeMemory());

delay(50);
}

```

- Breve spiegazione
 - Il codice sfrutta due strutture dati chiamate "function" e "relay". La struttura "function" ha il compito di contenere le informazioni che vengono ricevute con i pacchetti Bluetooth, mentre, la struttura "relay" ha il compito di contenere le informazioni relative alle prese del dispositivo fisico CENTRALINO. Ogni struttura "relay" avrà un suo orario di accensione e spegnimento, un suo orario per il timer e avrà una descrizione numerica che servirà invece a far capire all'utente che controlla CENTRALINO su quale "relay" andrà ad impartire un comando e soprattutto l'indirizzo del pin tramite il quale il microcontrollore sa dove inviare il segnale di accensione / spegnimento. Nel paragrafo seguente verrà spiegato perché è stato necessario adottare tante strutture "relay" quante sono le prese.

- Il sistema di comunicazione è molto importante perché grazie ad esso si riesce ad ottenere un livello di comunicazione all'interno del livello 7 del modello ISO/OSI il Bluetooth infatti copre tutti e 7 i livelli dell'ISO/OSI ma senza un'ulteriore divisione interna sarebbe stato impossibile avere più di una sola presa di corrente allo stesso CENTRALINO (rendendolo poco flessibile). Ogni pacchetto bluetooth è composto di 16 byte e ha un iniziatore di pacchetto "[" (ASCII: 91) e un terminatore "]" (ASCII: 93) senza di essi non si potrebbe determinare quando un pacchetto inizia o finisce e questo rende molto facile maneggiare i flussi di byte. All'interno del pacchetto ci sono ulteriori divisioni date dal simbolo "," (ASCII: 44) esso serve a capire quando inizia un settore del pacchetto e ne inizia un altro, in più danno un maggiore grado di integrità al pacchetto stesso (in caso di trasmissioni errate se non sono presenti "," il pacchetto non verrà considerato valido. Per maggiore comprensione guarda la funzione void bluetooth_Parser()). La struttura finale del pacchetto è questa.

Struttura di un pacchetto Bluetooth :

[Activity (2 cifre)	,	Extra_value (4 cifre)	,	Extra_value2 (6 cifre)]
---	-----------------------	---	--------------------------	---	---------------------------	---

Tabella 5

- Dentro ai pacchetti si trovano i comandi impartiti dall'applicazione software CENTRALINO. Activity varia in base Sveglia/Timer e Acceso/spento. Extra_value contiene il numero del pin a cui deve essere indirizzata il comando. Extra_value2 contiene nel caso della "Sveglia" il tempo in formato hh:mm:ss e nel caso del "Timer" il tempo in millesimi di secondo.
- Il sistema ha anche un "activity handler"; esso serve ad amministrare i pacchetti una volta che sono stati "smistati" e capire quali azioni sono richieste a

CENTRALINO. Con la variabile "activity" mostratasi nel punto precedente si immagazzina il codice che corrisponde all'azione che CENTRALINO deve compiere (Es. 01 e 02 sono le "activity" che corrispondono ad Accensione e Spegnimento istantanei). In "Extra_value" si trova il dispositivo a cui il pacchetto sta facendo riferimento, perciò il valore che troviamo in rel[x].socket è di fatto il pin a cui il relay è collegato e quello a cui fa riferimento Extra_value (questo meccanismo rende indirizzabili i pacchetti ai singoli dispositivi, similmente al "MAC address" di un computer). In "Extra_value2" si trova il valore suppletivo che dà manforte alla "activity". In "Extra_value2" si trovano ad esempio i tempi per le "svegli" espressi così "001650"(per motivi di integrità sono sempre 6 cifre), questa appena scritta è una sveglia per le 16 e 50.

- Esempio pacchetto : "[03,0006,001650]". Esso imposta una sveglia di accensione alle ore 16 50 per il pin 6.

- Problematiche principali.

- Le problematiche principali sono state tante ma per comodità del lettore ne verranno trattate solo due tra tante elencate e non.
 - 1) Il microcontrollore non ha un orologio interno è perciò impossibile far sì che esso capisca da se che ore sono al momento delle attività. Questo ha fatto sì che fosse necessario un componente adibito proprio a quello (Real Time Clock module) tuttavia questo componente non ha funzioni di sveglia o timer era perciò necessario che il microcontrollore gestisse da se l'orario e le scadenze delle attività. Inoltre si aggiunge l'ulteriore problema che se due strutture "relay" avessero avuto lo stesso orario di scadenza e il

tempo di acquisizione dell'orario fosse stato troppo lento c'era possibilità che il sistema fosse troppo lento a gestire le sveglie. Si è risolto con la funzione `void turn_by_match()` che controlla ogni singola struttura "relay" in tempi molto al di sotto del secondo e fa sì che sia possibile spegnere/accendere più "relay" contemporaneamente anche quando gli orari combaciano al secondo.

- 2) Doveva essere possibile far sapere all'applicazione software CENTRALINO quali dispositivi erano connessi al dispositivo fisico CENTRALINO e per fare questo era necessaria una descrizione anche superficiale delle prese di corrente. Per questo è stata introdotta la variabile "description" che ha una leggenda (nel codice sotto il nome di MEANINGS OF DESCRIPTION). Il meccanismo tramite il quale l'applicazione viene a sapere le varie "description" delle strutture "relay" è questo :
 - App. Software CENTRALINO manda pacchetto con activity = 50 ovvero una Discovery (richiede che gli vengano mandate le informazioni).
 - Disp. Fisico CENTRALINO manda tanti pacchetti quante sono le strutture "relay" con activity = 51 e la relativa `rel[].description` (description di ogni struttura) e infine manda un pacchetto con activity = 99 che significa che i relay sono terminati e l' A.S. CENTRALINO può smettere di aspettare che glie ne arrivino altri.
- 3) Zero padding (non affrontata sul documento).

- 4) Conflitti di formato dati e atoi() (non affrontata sul documento).

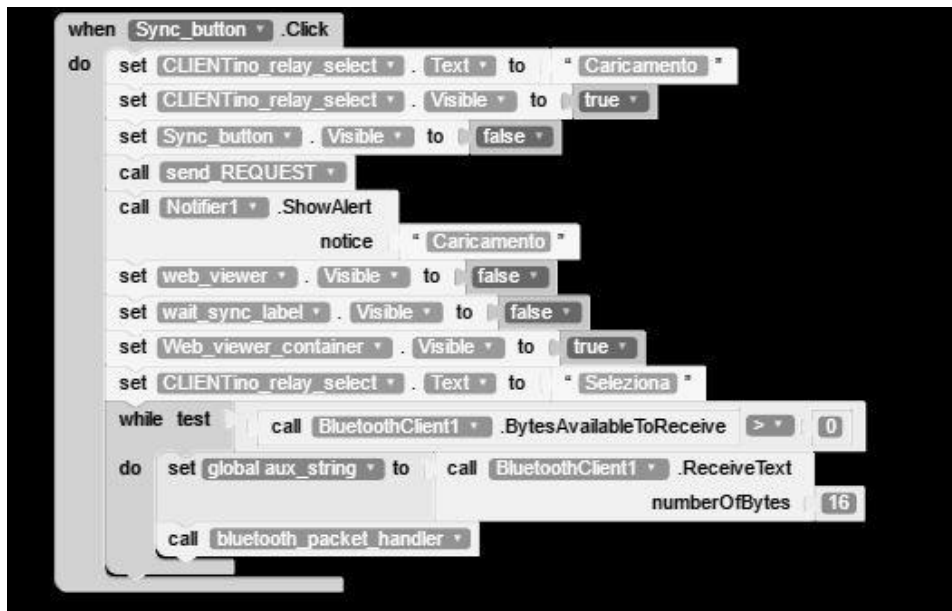
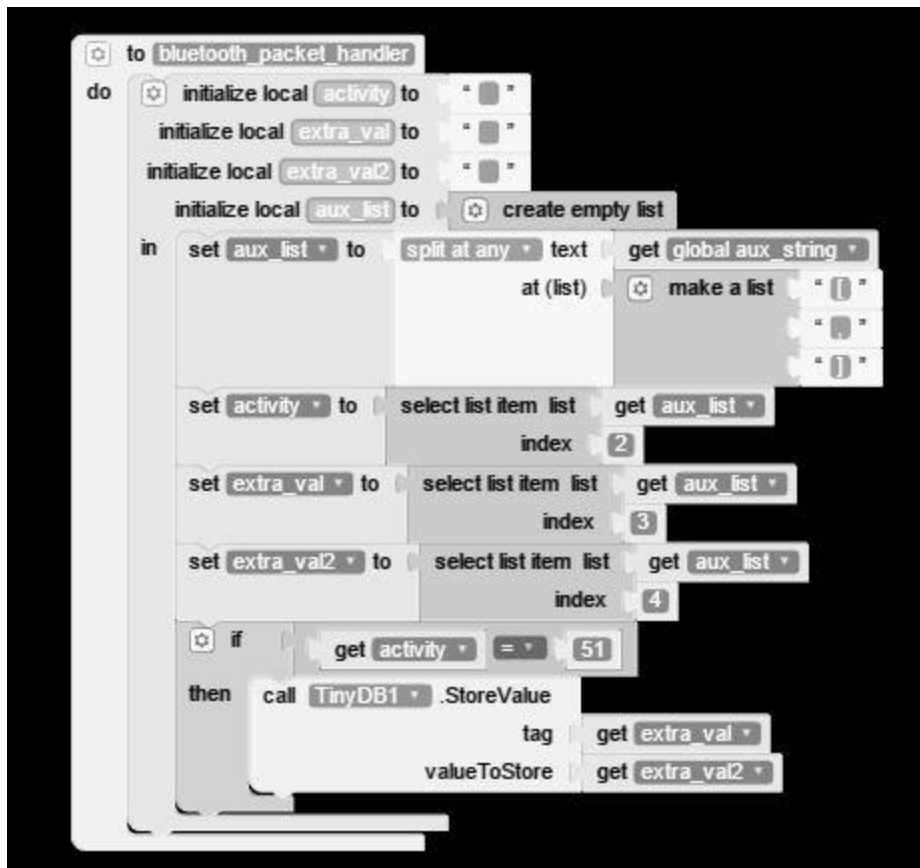
Codice applicazione CENTRALINO

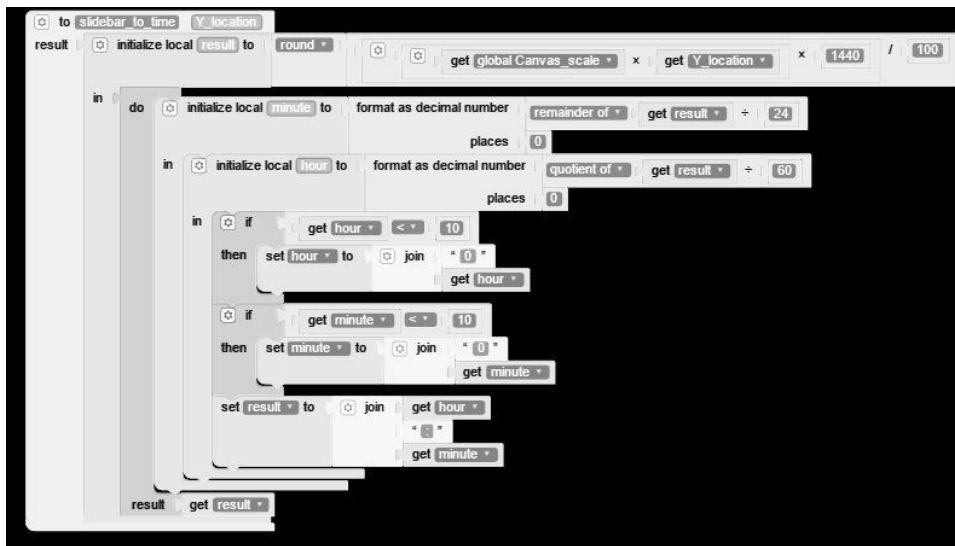
- Codice descrittivo della grafica dell'applicazione.

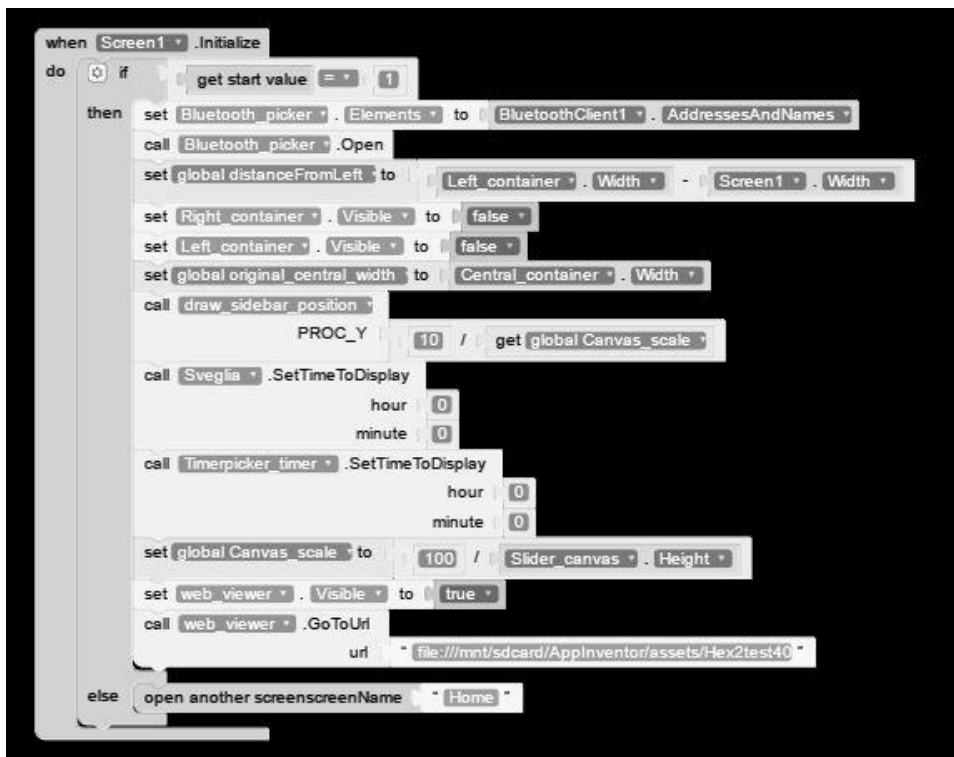
{I seguenti documenti si trovano nel file .aia generato dal framework di App Inventor}

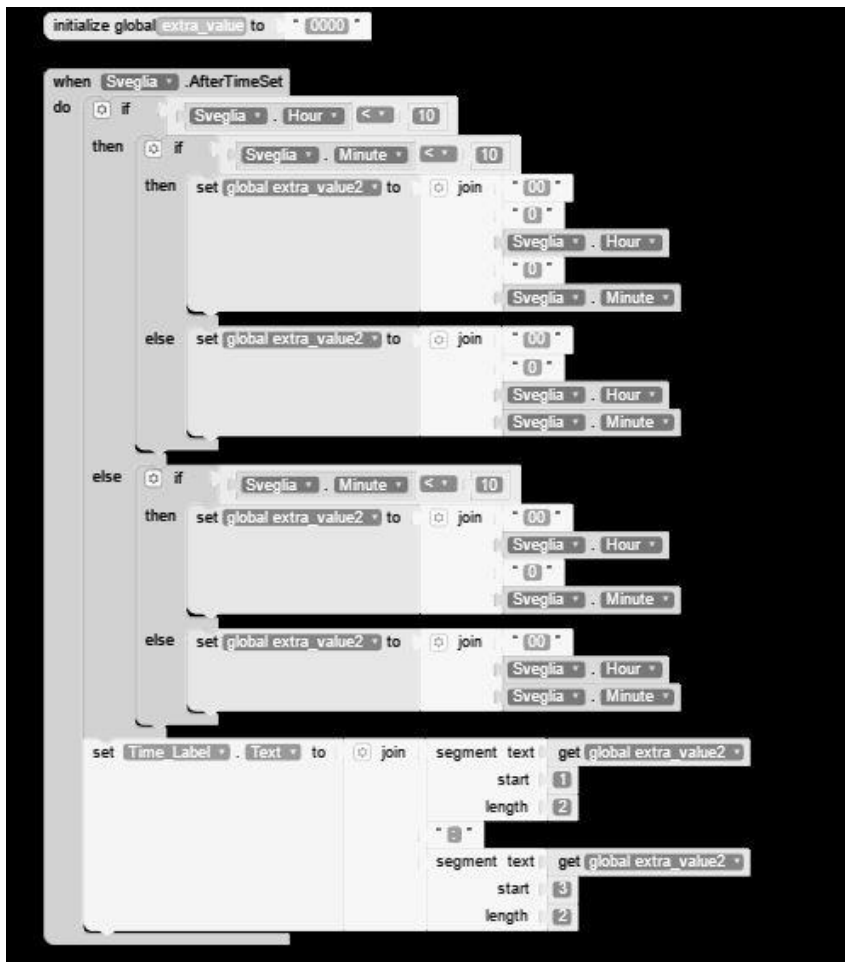
- Codice operativo dell'applicazione.

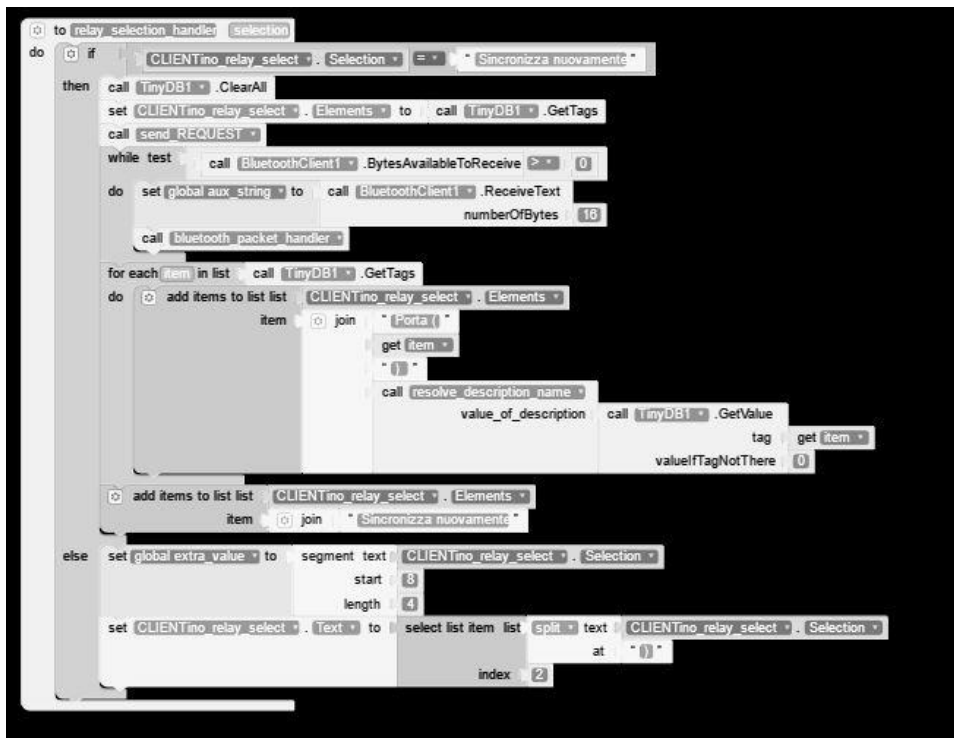
Per motivi di compatibilità questa parte di codice avrà un aspetto leggermente differente da quello visto fino ad ora. Essendo App Inventor ancora in una versione BETA non è possibile esportare i progetti in formato testuale perciò si è costretti ad esportarlo in formato grafico.

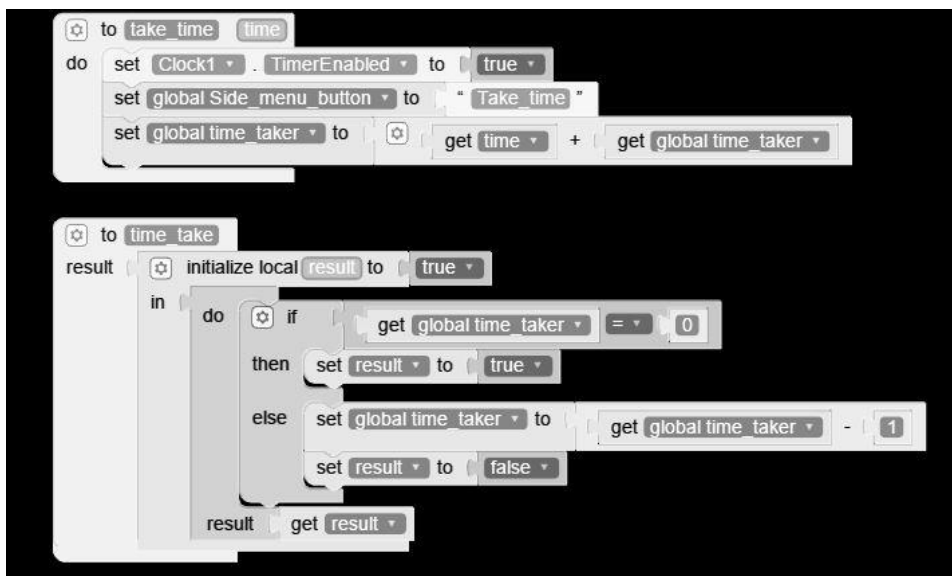
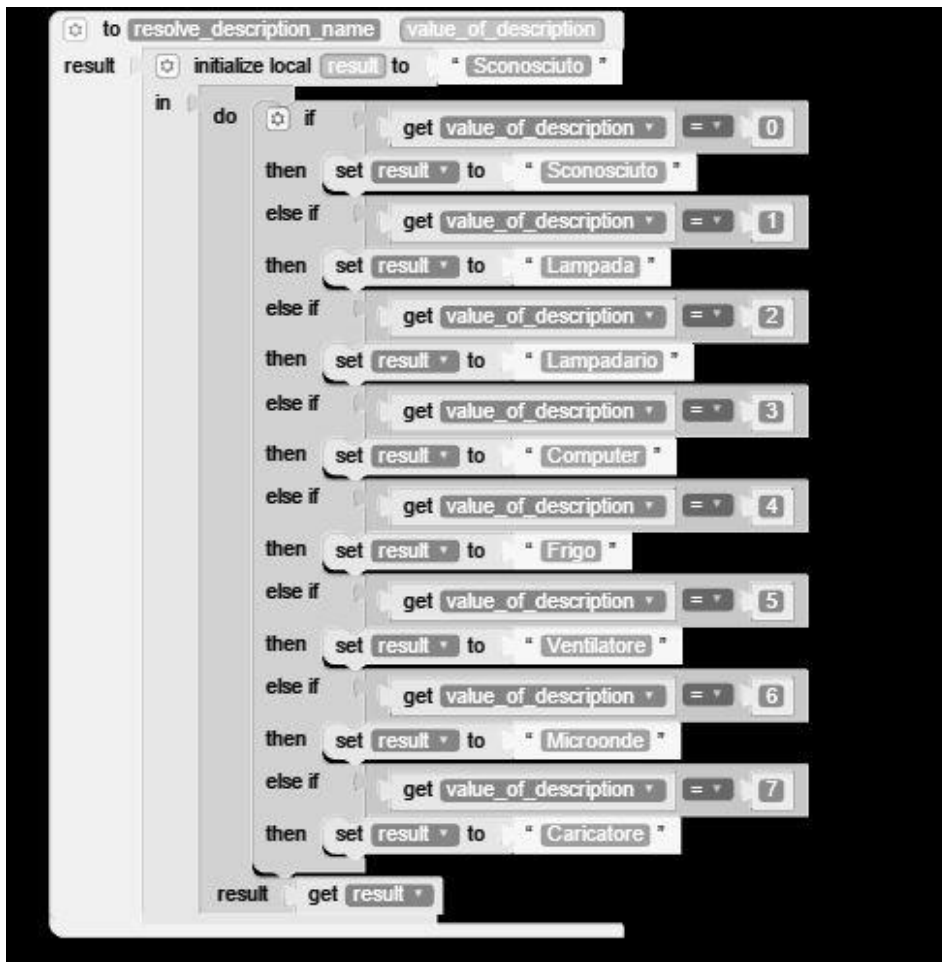


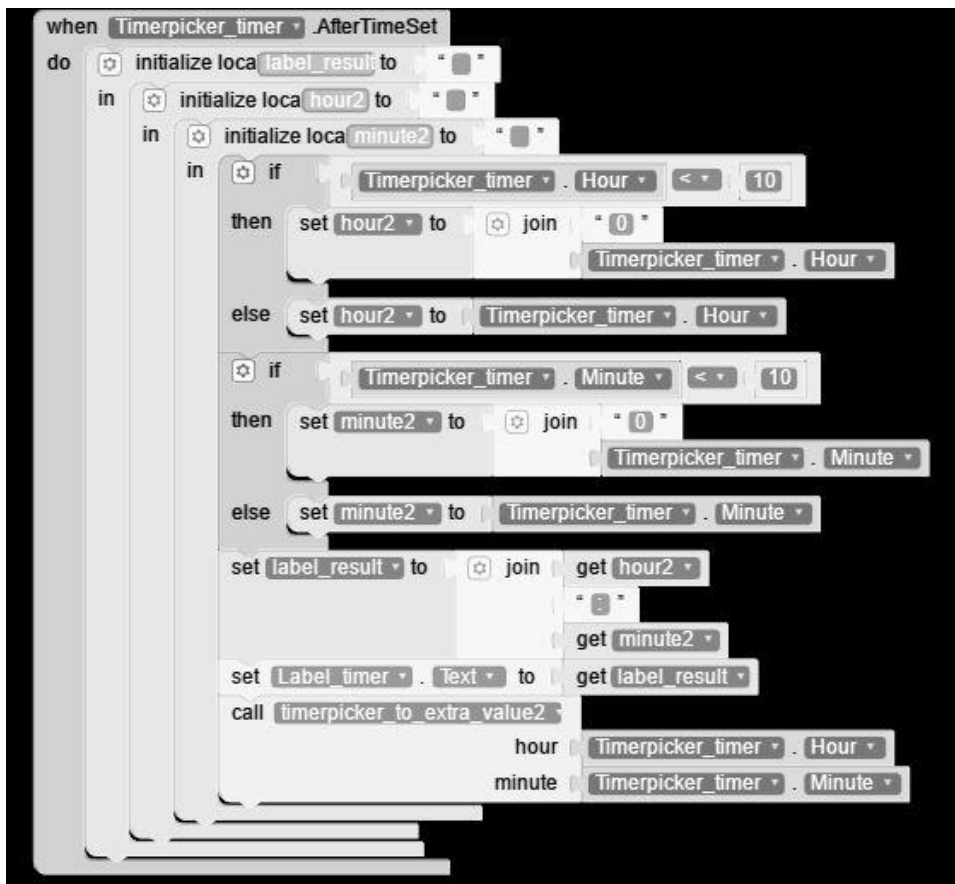
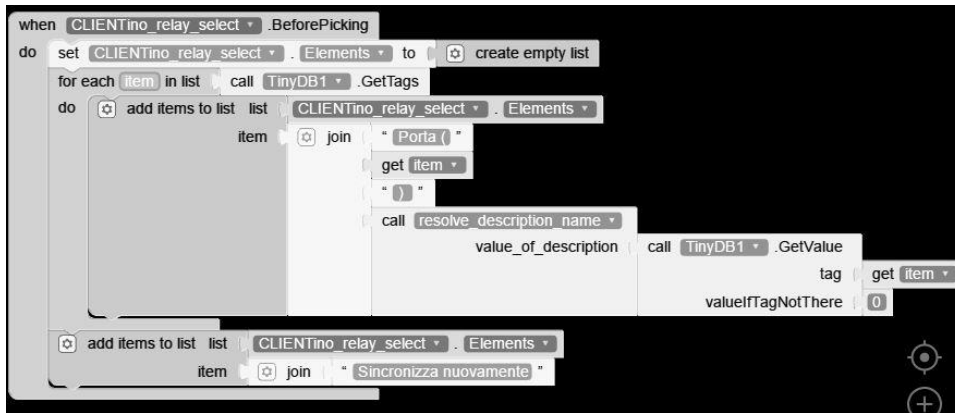


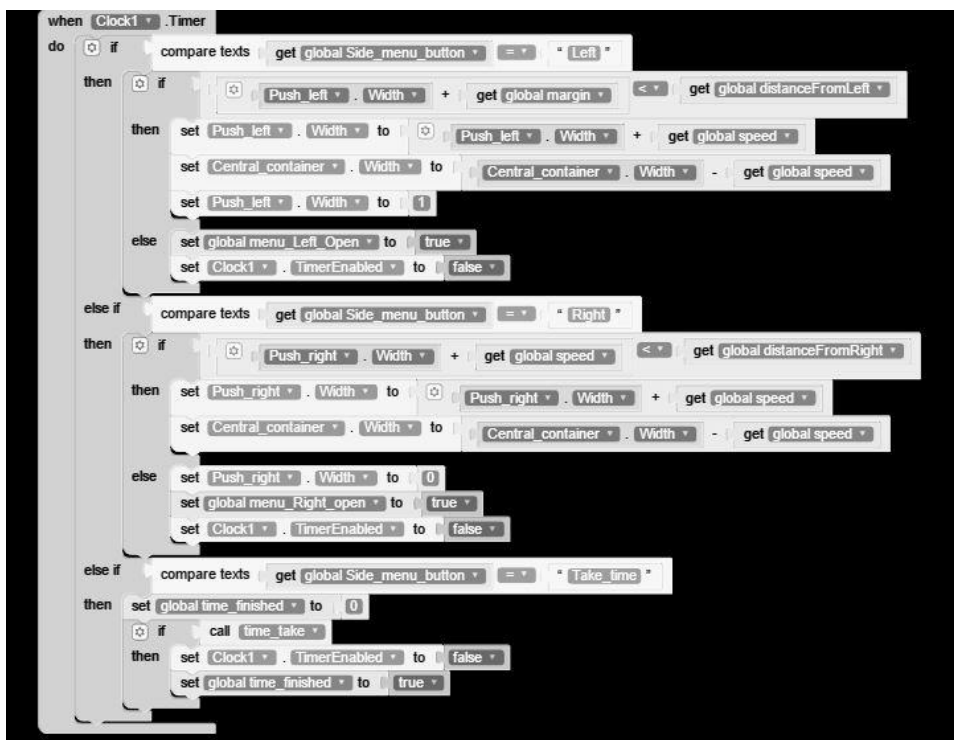
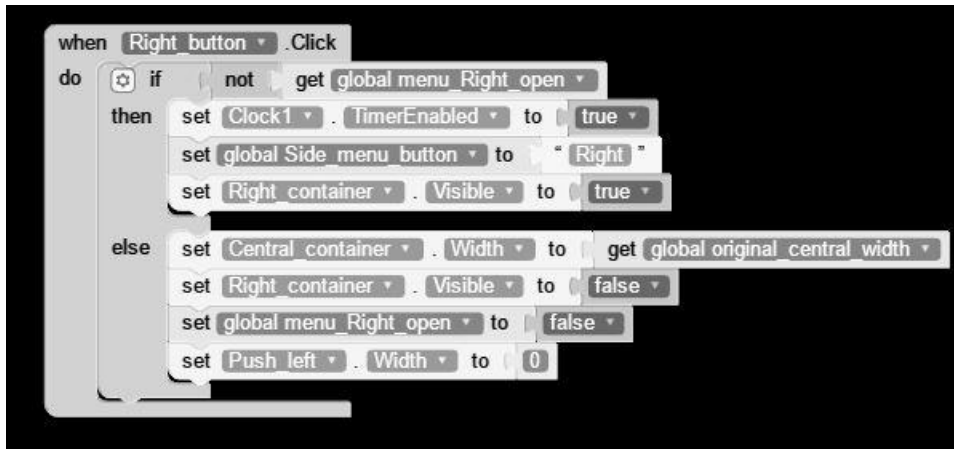


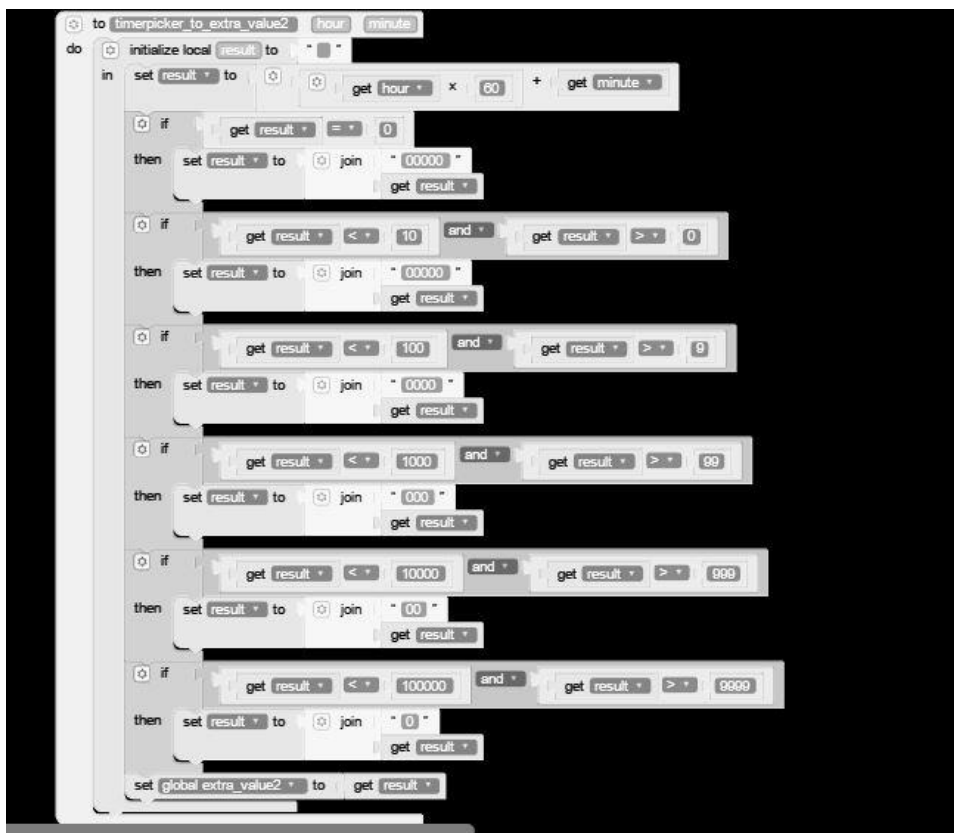
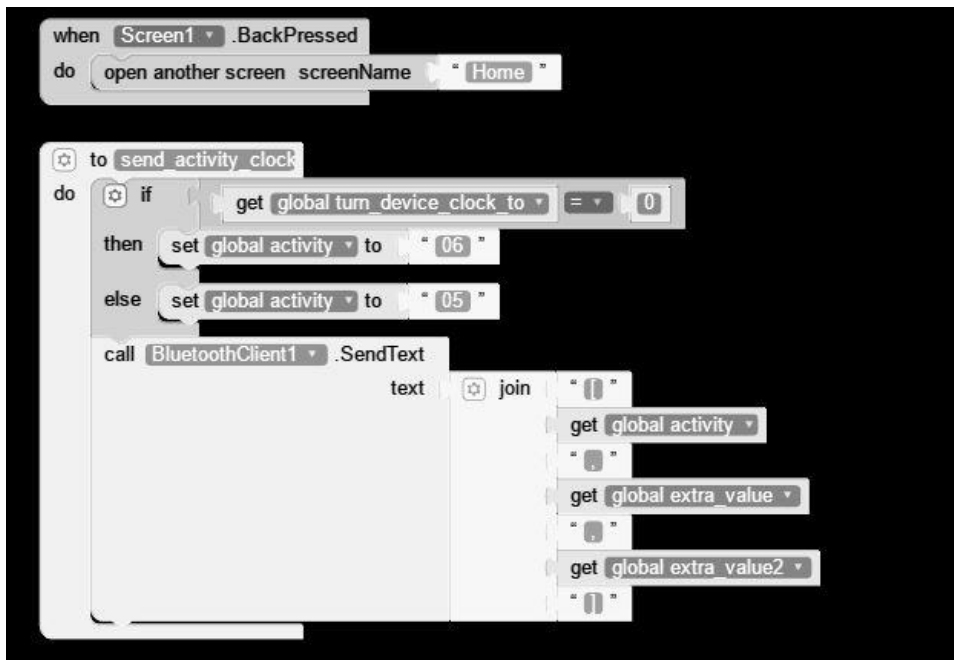


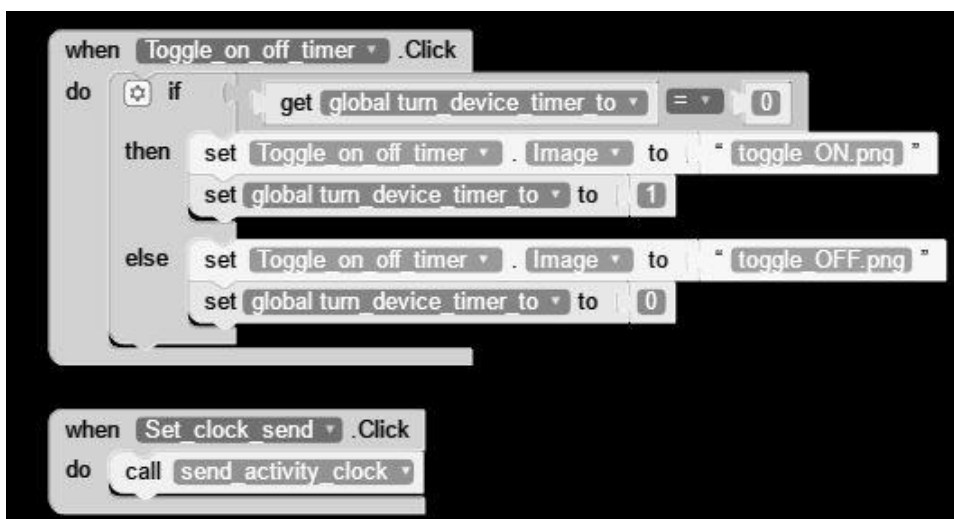
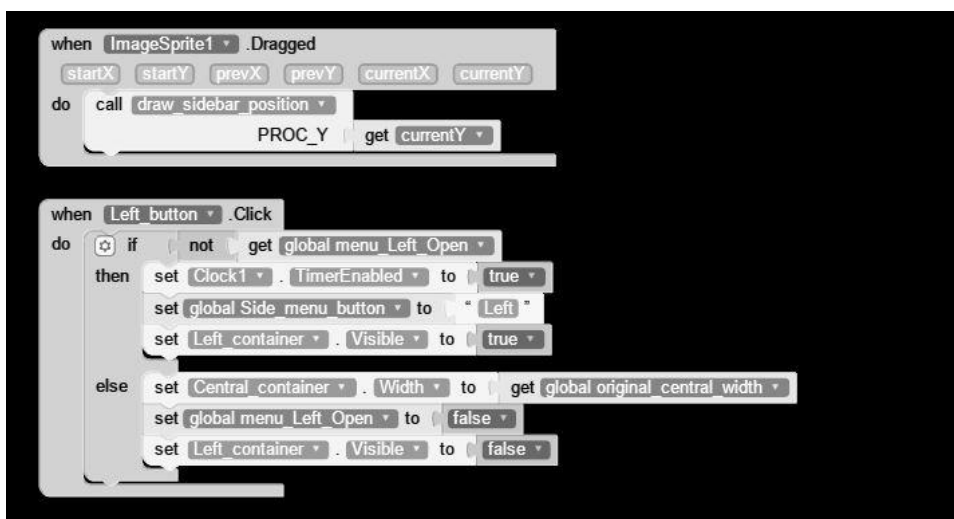
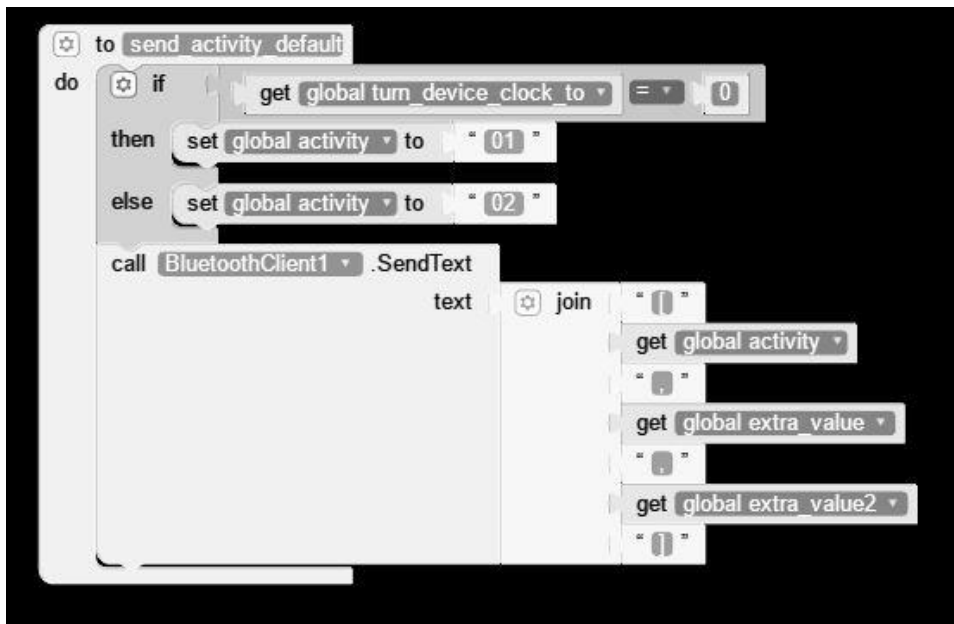


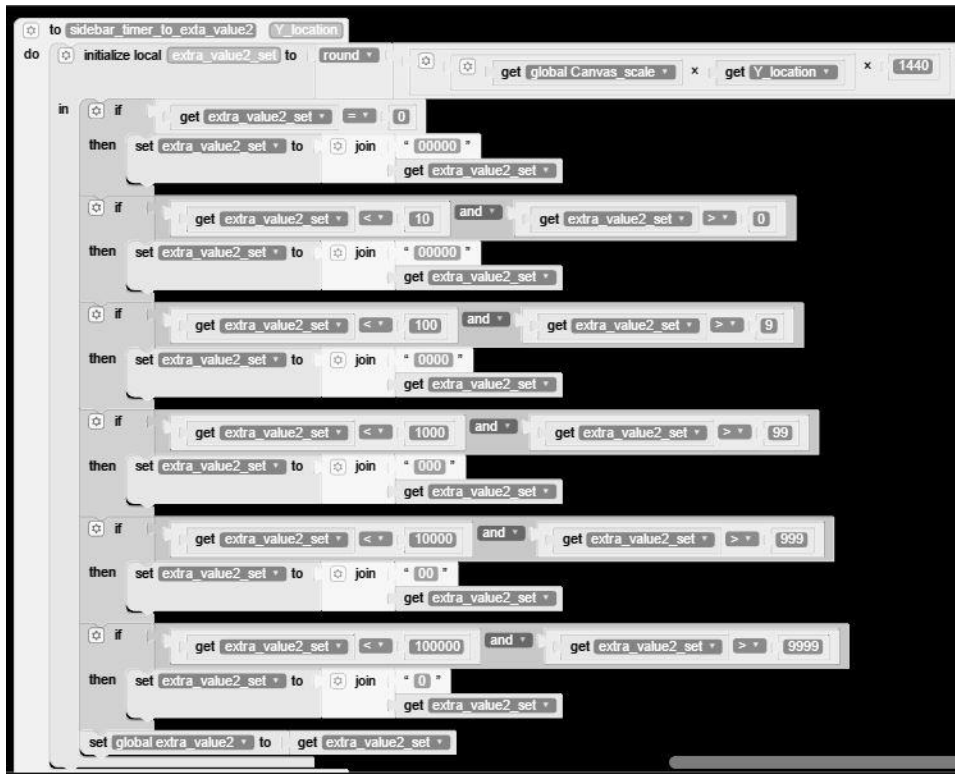












- Breve spiegazione
 - L'applicazione sfrutta principalmente l'utilizzo di variabili globali che vengono continuamente usate in blocco per mandare i pacchetti con le "activity". Queste tre variabili prendono il nome dal sistema del dispositivo fisico CENTRALINO, pertanto si chiamano anche esse "activity", "extra_value" e "extra_value2".
 - La piattaforma di sviluppo di App Inventor si basa sull'utilizzo degli eventi. Gli eventi di Android fungono come un kernel di sistema. Essi permettono di sapere sempre quando una determinata azione è accaduta e permettono quindi di

richiamare funzioni specifiche quando quegli "eventi" si verificano. Per esempio quando un pulsante viene premuto solleverà l'evento `pulsante.onclick()` e sarà possibile chiamare una propria "procedure" (funzione) ogni qualvolta si solleverà l'evento. Gli eventi sono molti e molto variegati. Sono stati utilizzati principalmente gli eventi legati allo schermo ovvero `"screen.BackPressed()"` e `"screen.Initialize()"`, gli eventi legati ai pulsanti anche di vario tipo (`"timePicker"` e simili) e come si può evincere dal codice operativo si è fatto largo uso dell'oggetto/componente `tinyDB` con i suoi vari metodi e funzioni.

○

- Problematiche principali

- Innanzitutto il passaggio da un sistema di programmazione tradizionale ad uno più articolato può essere fuorviante. Certamente lo è stato, iniziare a ragionare non più per flusso di codice ma per eventi. Sapere che il codice non fa sempre le stesse operazioni in sequenza bensì si comporta in modo differente in base agli eventi che nascono è molto differente da ciò che è stato appreso in classe, tuttavia con il tempo è stato possibile districarsi sempre meglio in questo tipo di mentalità di programmazione.
- Non essendo App Inventor un vero e proprio linguaggio di programmazione è molto raro trovare documentazione affidabile sulla sintassi dei comandi, fortunatamente il loro funzionamento è molto intuitivo ma nei casi in cui questa intuitività è venuta meno, c'è stato bisogno di spendere molto tempo nella ricerca e nella comprensione di comandi adeguati al perseguimento degli obiettivi.

- Un grande problema è stato riscontrato nell'utilizzo stesso di App Inventor proprio per via del fatto che è una piattaforma in via di sviluppo e si trova ancora nella fase BETA, è spesso soggetta a crash, bug di varia natura e frequenti interruzioni di funzionamento parziale.
- Conclusioni :
 - Futuri sviluppi:
 - CENTRALINO come già detto è un progetto che ha ancora molte potenzialità da sviluppare, anzi , si può dire quasi che questo primo lavoro sia solo l'inizio. In futuro si prediligerà certamente un altro tipo di integrazione con l'ambiente, come ad esempio, macchinari che nascono di fabbrica con la possibilità di integrarsi in sistemi domotici.
 - Dedicato
 - Dedico questo lavoro ai professori e i compagni che mi hanno spinto a fare questo progetto, che hanno creduto che lo potessi fare e mi hanno sorriso quando cercavo di spiegare cosa fosse quando ancora non aveva forma. Ancora di più lo dedico e ringrazio i professori e i compagni che non hanno creduto che lo potessi fare, e si sono stupiti quando il progetto ha iniziato a prendere forma, senza l'una o l'altra non avrei avuto la spinta necessaria a portarlo a termine.

Fonti

" https://it.wikipedia.org/wiki/Licenza_MIT "

" [https://it.wikipedia.org/wiki/Costruzionismo_\(teoria_dell'apprendimento\)](https://it.wikipedia.org/wiki/Costruzionismo_(teoria_dell'apprendimento)) "

" <https://it.wikipedia.org/wiki/Domotica> "