Word Clock

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Pianificazione 5

1.7 Analisi dei mezzi 6

1.7.1 Software 6

1.7.2 Hardware 6

2 Progettazione 7

2.1 Analisi e verifica funzionamento componenti 7

2.1.1 Striscia di led 7

2.1.2 RTC 8

2.2 Design dell’architettura del sistema 9

2.3 Design delle interfacce 10

3 Implementazione 11

4 Test 12

4.1 Protocollo di test 12

4.2 Risultati test 13

4.3 Mancanze/limitazioni conosciute 13

5 Consuntivo 14

6 Conclusioni 15

6.1 Sviluppi futuri 15

6.2 Considerazioni personali 15

7 Bibliografia 16

7.1 Bibliografia per libri 16

7.2 Sitografia 16

8 Allegati 16

# Introduzione

## Informazioni sul progetto

* Allievi: Gabriele Alessi, Mattia Lazzaroni, Paolo Claudio Weishaupt.   
  Superiore professionale: Adriano Barchi.
* Scuola d’Arti e Mestieri di Trevano, sezione Informatica, classe 3, modulo 306.
* Data inizio: 13.02.2019.  
  Data fine: 22.05.2019.

## Abstract

*A word clock is a clock that shows the time by using spoken language. In this project a box is used where below there is a strip of led and above there is a sheet with the words that together form the time. This strip is programmed in Arduino so that the led that form the words of the current time are turned on. The time is obtained from a time server and set up by a RTC, otherwise it can be configured directly from the box using physical button.   
The final product can be considered as a prototype, since the main idea is to expose a large word clock on the roof of the study center.*

## Scopo

Lo scopo di questo progetto è quello di creare un prototipo funzionante di un orologio a parole (word clock) per poi in seguito presentarlo a una commissione che deciderà se stanziare i fondi per poterne realizzare una versione più grande montata sulla torre sinistra della scuola.   
Un word clock è un orologio che mostra l’orario tramite l’uso del linguaggio parlato. In questo progetto viene usata una scatola dove viene montata una striscia di led con sopra montato un sostegno in cui appoggiare un foglio dove sono scritte le parole. Questa striscia viene programmata in Arduino, in questo modo i led si accendono formando le parole che indicano l’orario. L’orario è ottenuto da un server e viene poi impostato e memorizzato da un RTC, altrimenti è possibile configurarlo direttamente dalla scatola tramite pulsanti fisici.

# Analisi

## Analisi del dominio

L’entrata del quarto piano è poco invitante e non ci sono esposti dei progetti realizzati dagli allievi.   
Quindi questo prodotto colmerebbe questo spazio vuoto mostrando qualcosa di utile (l’orario) in maniera diversa e interessante. Il contesto in cui il prodotto dovrà funzionare è adatto e sarebbe l’ideale metterlo all’inizio del corridoio. Attualmente esiste un piccolo orologio binario nella zona degli elettronici, ma questa soluzione è più chiara e leggibile, quindi non ci sarebbero problemi nel metterlo in mostra.  
Gli utenti sono principalmente gli allievi del quarto piano, che arrivando al corridoio verrebbero a conoscenza dell’orario solamente alzando lo sguardo e sapere se sono in ritardo per la lezione o meno. Quindi gli utenti del prodotto non necessitano e non necessiteranno di particolari competenze (come con l’orologio binario).

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Verifica componenti hardware |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | È necessario verificare che i componenti hardware (Fishino e led) funzionino correttamente. |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Implementare il Word Clock |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | È necessario implementare il Word Clock in modo che mostri l’orario sotto forma di linguaggio parlato. |
| **Sotto requisiti** | |
| **001** | Scrivere il testo con le differenze di 5 minuti |
| **002** | Mostrare i secondi tramite l’utilizzo di 12 led, quindi 5 secondi per ogni pallino acceso. |
| **003** | Si dovrà poter controllare e impostare l’orario e le impostazione tramite il modello fisico. |
| **004** | Cambiare i colori dei led in base all’orario. |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Controllo via web |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Si dovrà poter controllare e impostare l’orario e le impostazione tramite una pagina web. |

## Pianificazione

La pianificazione del progetto è stata effettuata mediante la realizzazione di un diagramma di Gantt.



## Analisi dei mezzi

### Software

I software utilizzati per la realizzazione di questo progetto sono:

* Microsoft Word 2016
* Microsoft Power Point 2016
* Arduino IDE 1.8.7
* GanttProject 2.8.9 Pilsen
* Google Chrome 71.0.3578.98
* Atom 1.34.0
* FishinoFlasher 5.0.0

### Hardware

* Computer Mattia Lazzaroni:
  + Modello: Acer Aspire E 15
  + Processore: Intel Core i7 7500u
  + RAM: 16GB LPDDR4 2133 MHz
  + GPU: NVIDIA 940MX
  + SSD: 256 GB
* Computer Gabriele Alessi:
  + Modello: HP ENVY Notebook,
  + Processore: Intel Core i7
  + RAM: 16GB
* Computer Paolo Weishaupt:
  + Modello: Huawei MateBook X Pro
  + Processore: Intel Core i7-8550U
  + RAM: 8GB
  + Display 13.9” 3000x2000px
  + SSD: 512 GB
* Fishino UNO REV2:
  + Compatibile al 100% con Arduino UNO
  + Modulo WiFi integrato
  + Slot per schede MicroSD integrato
  + Modulo RTC integrato
  + Connettore sfalsato per facilitare l’uso con breadboards
  + Sezione di alimentazione a 3.3V potenziata
  + Compatibilità con le schede millefori
  + Schemi elettrici, file Eagle e pinout della scheda
* Adafruit NeoPixel
  + 195 led RGB
* Arduino Mega 2560
* Componenti:
  + Resistenza 500Ω
  + Condensatore 1000µF
  + Alimentatore di supporto

# Progettazione

In questo capitolo viene spiegato come è stato ideato il prodotto e come è stata preparata la fase di implementazione. Inizialmente è stato organizzato l’ambiente di sviluppo (software, striscia di led, resistenze). Poi sono stati analizzati i componenti del progetto e che funzionassero correttamente, provando a eseguire qualche programma di test cambiano i parametri.   
Successivamente è stata fatta un’idea della struttura del sistema, definendo i file e la struttura delle cartelle. Infine si è pensato a come il prodotto si sarebbe effettivamente presentato, procurandoci il materiale necessario per mostrare le parole (fogli e protezioni).

## Installazione ambiente di sviluppo

Prima di iniziare a lavorare è stato necessario impostare un ambiente comune ai membri del team e definire da dove cominciare. Quindi si è deciso di studiare i moduli da sviluppare così da identificare i vari componenti che sarebbero stati utili per la realizzazione del prodotto.

### Librerie

* Librerie Adafruit
  + Adafruit NeoPixel

Libreria che presenta funzioni basate per il controllo di strisce di led RGB prodotte da Adafruit. Nella libreria ci sono funzionalità che sono necessarie per lo sviluppo del progetto, come ad esempio l’accensione di un led di un certo colore.  
*Link:* <https://github.com/adafruit/Adafruit_NeoPixel>

* + RTClib

Gestione di un dispositivo RTC in un ambiente di sviluppo Arduino. La libreria presenta metodi utili per ottenere l’orario corrente e tutti i suoi componenti (anno, mese, giorno, ore, minuti, secondi).   
*Link:* <https://github.com/adafruit/rtclib>

* Librerie Arduino e Fishino
  + WiFi

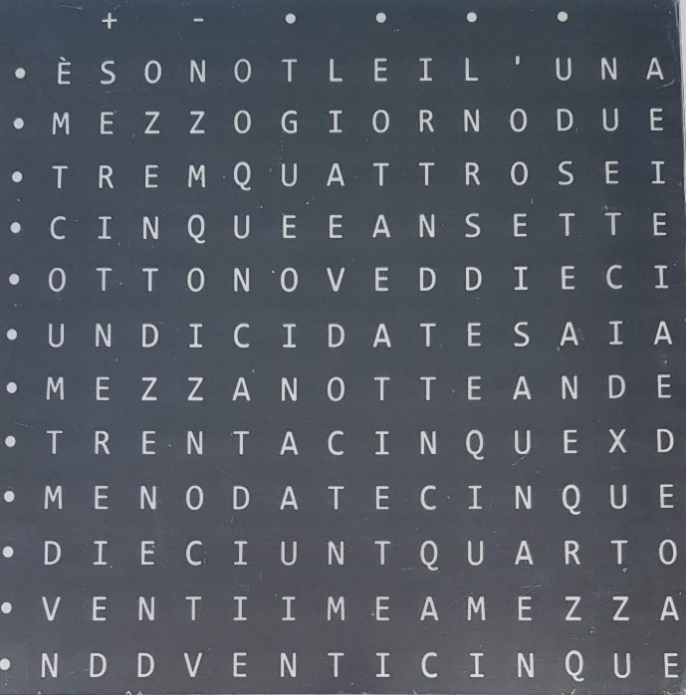
Grazie a questa libreria, il dispositivo sarà in grado di connettersi a internet. Ciò serve per potersi connettere a un server e ricevere l’orario via wireless.   
*Link:* <https://www.arduino.cc/en/Reference/WiFi>

* + Fishino

Se si usa un Fishino è necessario scaricare la libreria dal sito ufficiale per integrarla in Arduino IDE. Questa libreria è essenziale per il funzionamento del dispositivo e integrare i vari moduli come WiFi e RTC.  
*Link:* <https://www.fishino.it/download-libraries-it.html>

### Scatola del word clock

Il “corpo” del prodotto comprende una scatola di legno dove viene piazzata la striscia di led e sopra di essa viene messo un sostegno che separa ogni led dagli altri.





## Analisi e verifica funzionamento componenti

### Striscia di led

Per poter sviluppare il prodotto è stato inizialmente necessario verificare che la striscia di led funzionasse correttamente. Per fare ciò abbiamo usato un Arduino Mega 2560 e abbiamo caricato il programma *strandtest* della libreria di Adafruit. Però non siamo stati attenti e abbiamo avviato il test senza seguire le istruzioni del programma e della guida, le quali affermano che è necessario disporre di una resistenza sul pin dei dati e di un condensatore per avere una situazione sicura in caso di problemi con la corrente.   
Quindi dopo alcuni test e cambiando dei parametri siamo giunti alla conclusione che i led funzionassero tutti regolarmente, tuttavia ci sono dei problemi quando si accendono tanti diodi con un colore luminoso (bianco), infatti il colore tende a spegnersi e il sistema a crollare forzando un riavvio. Ma tutto ciò si può risolvere con l’aiuto di un alimentatore in modo da dare alla striscia più corrente.



Questo è un metodo che accende tutti i led uno dopo l’altro. È stato utile per capire dopo quanti led accesi di bianco il sistema sarebbe saltato. Inoltre è stato scoperto anche che per definire il colore di un led bisogna usare il tipo di variabile *uint32\_t*, altrimenti alcuni colori non verrebbero indicati in modo giusto.

### RTC

Il Real Time Clock è un modulo integrato nel Fishino che permette di salvare e aggiornare le informazioni riguardanti data e ora. Per testare e imparare a usare questo componente sono stati creati dei metodi e fatte delle prove in un display LCD.



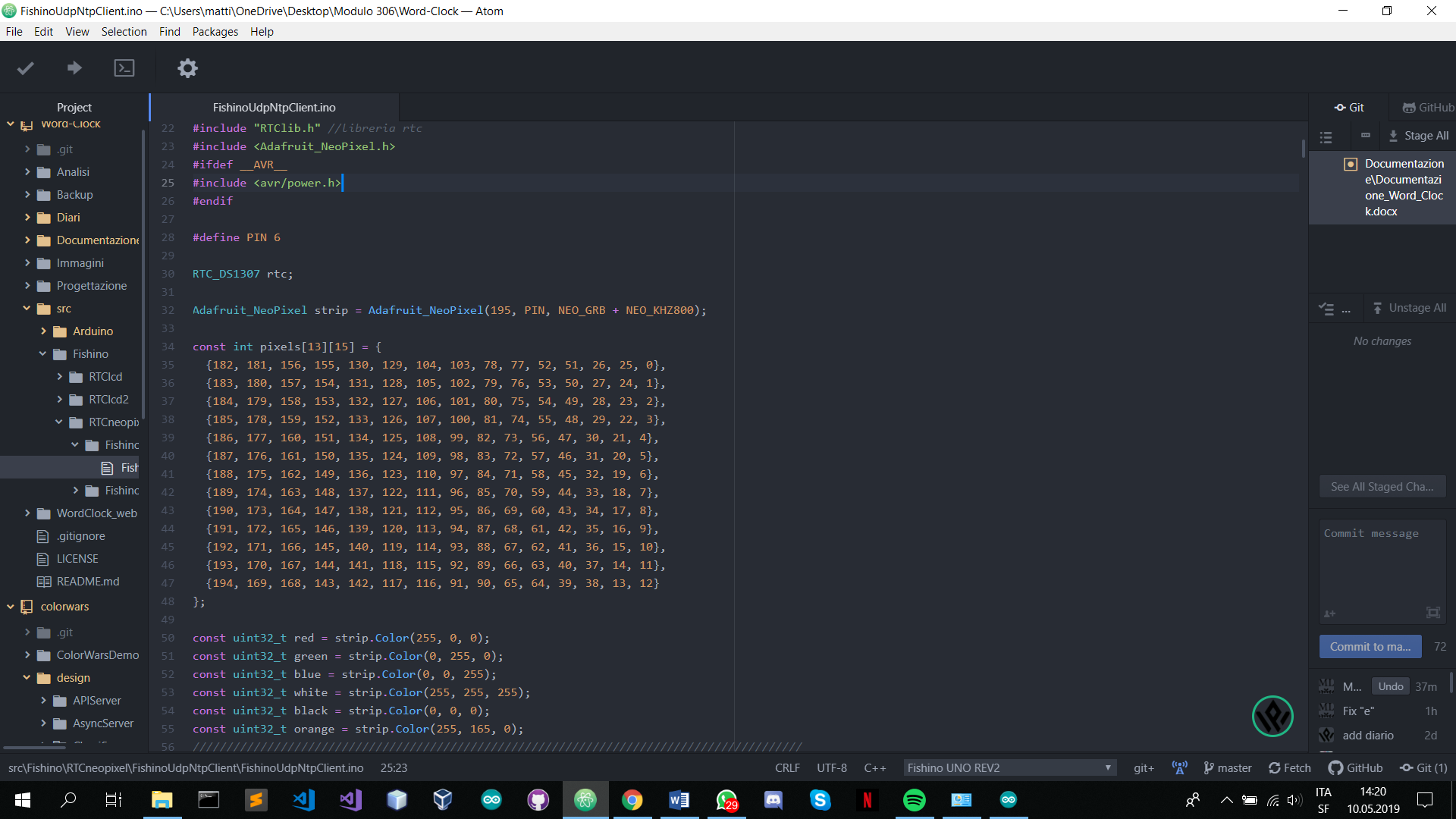
In questo test sono stati impostati LCD, RTC e un array per indicare i giorni della settimana.

## Design dell’architettura del sistema

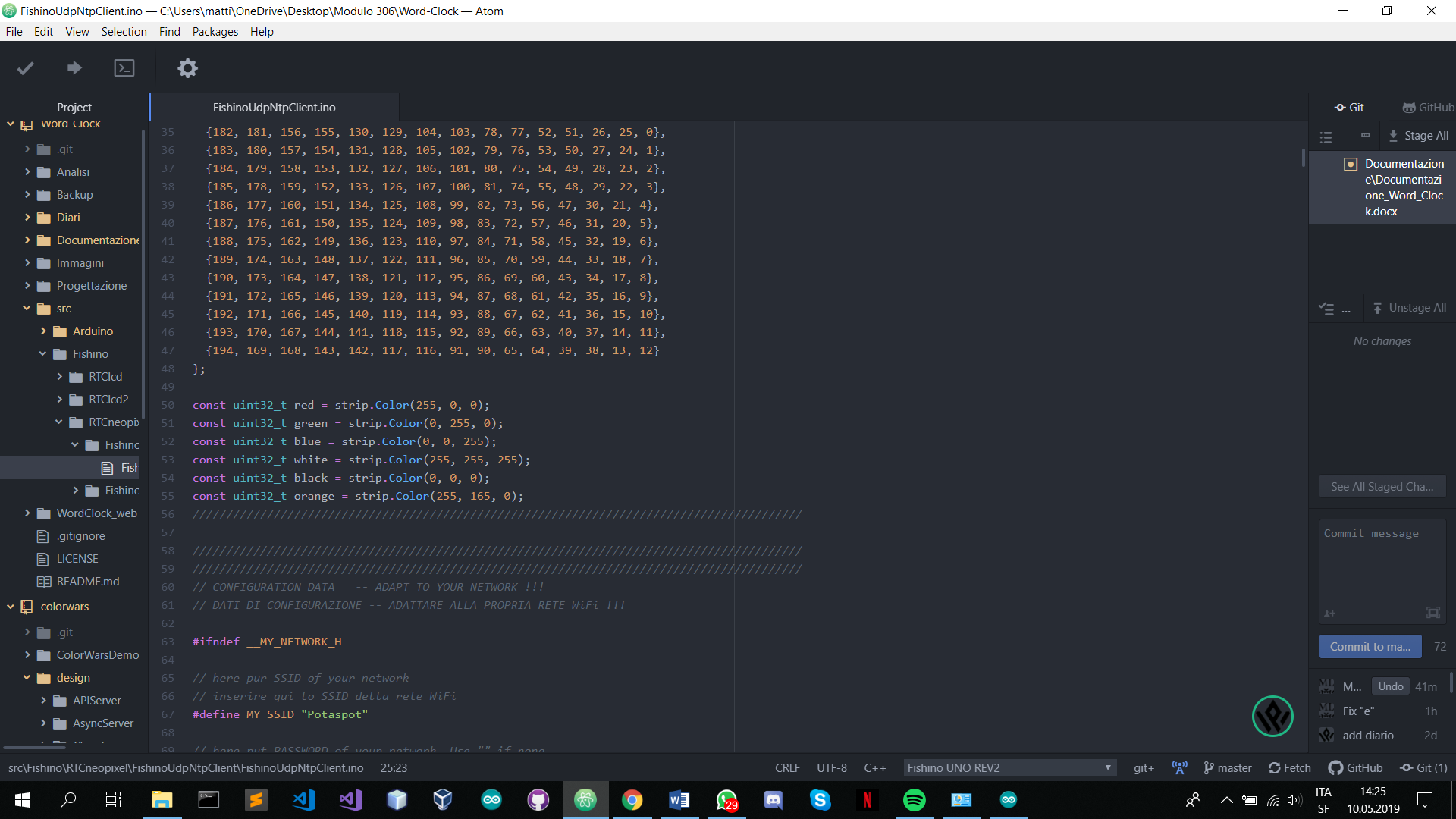
## Design delle interfacce

# Implementazione

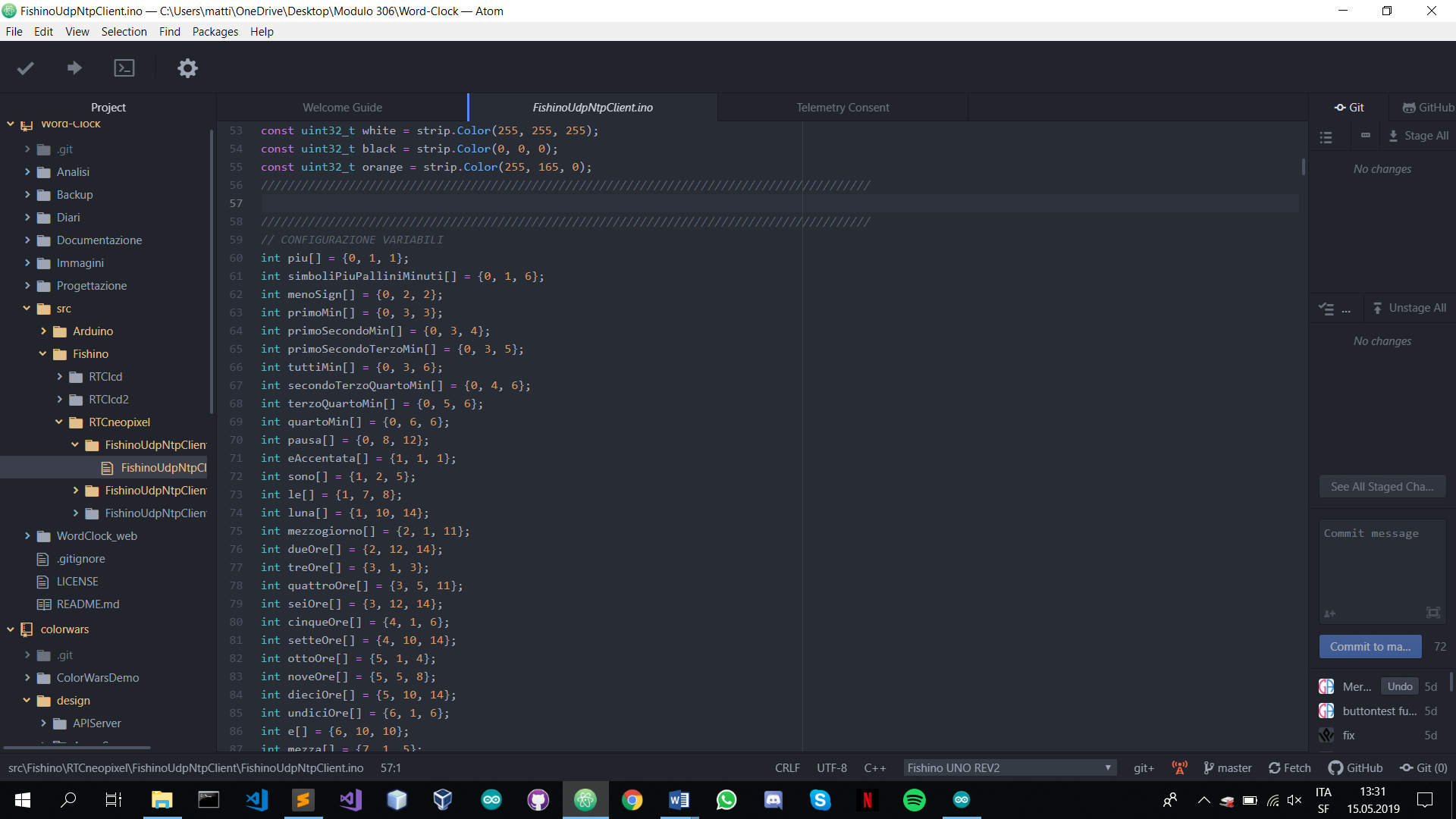
Inizialmente abbiamo creato un array bidimensionale di interi per rappresentare la matrice di led. Questo è dovuto al fatto che abbiamo deciso di non mettere lo 0 in alto a sinistra, bensì in alto a destra, per poi continuare verso il basso e da destra verso sinistra. I numeri rappresentano l’ordine di accensione dei led: più il numero dell’array è basso, prima si accenderà.



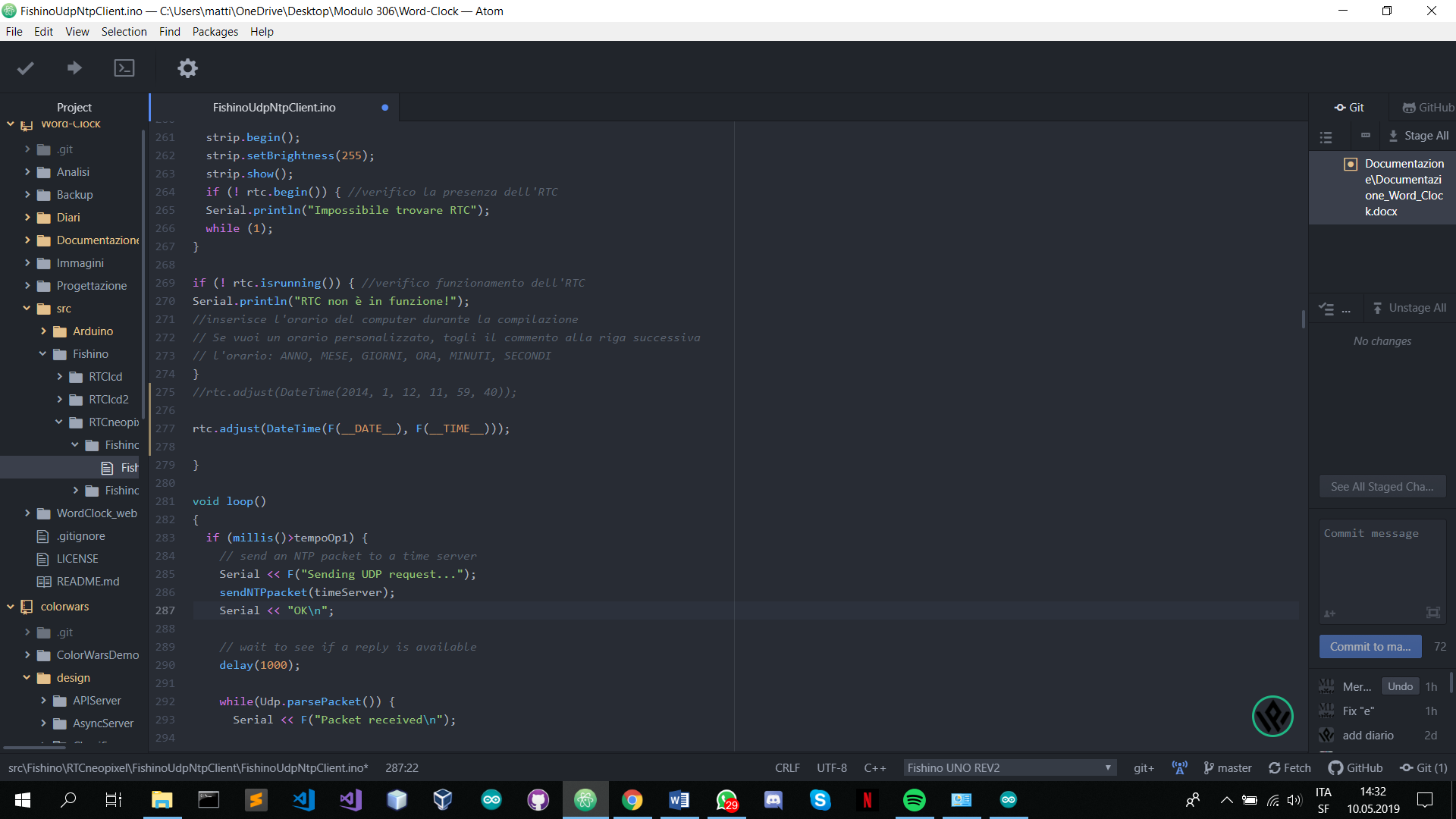
Successivamente abbiamo dichiarato tutte le costanti che rappresentano il colore (in RGB) della striscia.



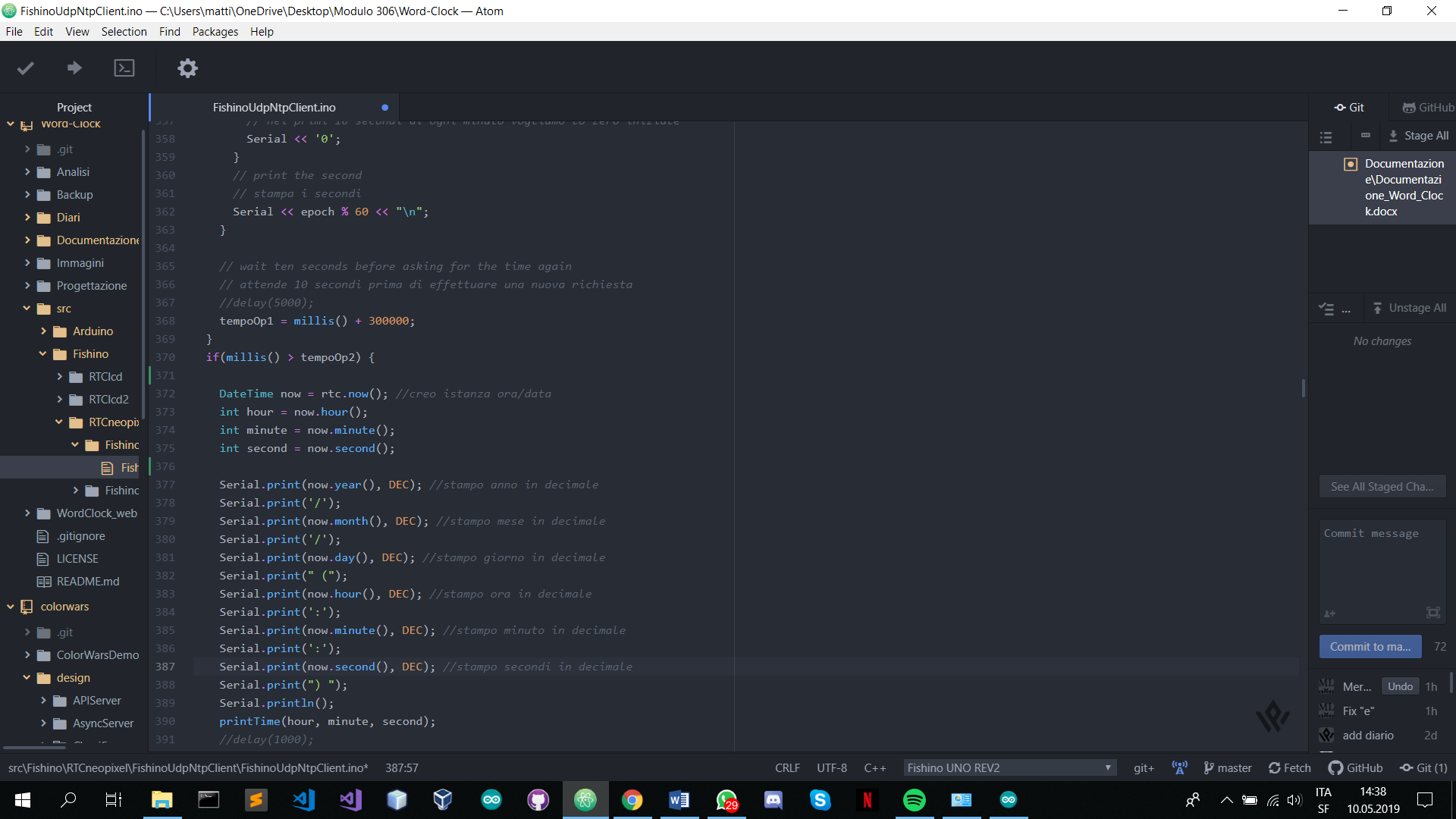
In seguito, è possibile vedere una parte delle innumerevoli variabili che abbiamo dichiarato. Ognuna di queste variabili è un array di interi e rappresenta una scritta da fare accendere. In questi array come primo valore immettiamo il numero della riga in cui c’è la scritta da accendere. Come secondo e terzo valore specifichiamo il range dei led da accendere orizzontalmente. Ad esempio, per il segno “+”, abbiamo dichiarato la variabile “piu[]”. Nell’array, alla prima posizione troviamo uno 0, che indica appunto che il segno “+” si trova nella prima riga. Nella seconda e nella terza posizione notiamo due “1”. Indicano che il “+” si trova nella seconda colonna a partire da sinistra (e non nella prima, dato che lo 0 rappresenta la prima colonna).



Nel metodo Setup() con la seguente riga di codice impostiamo la data e l’ora corrente sull'RTC:

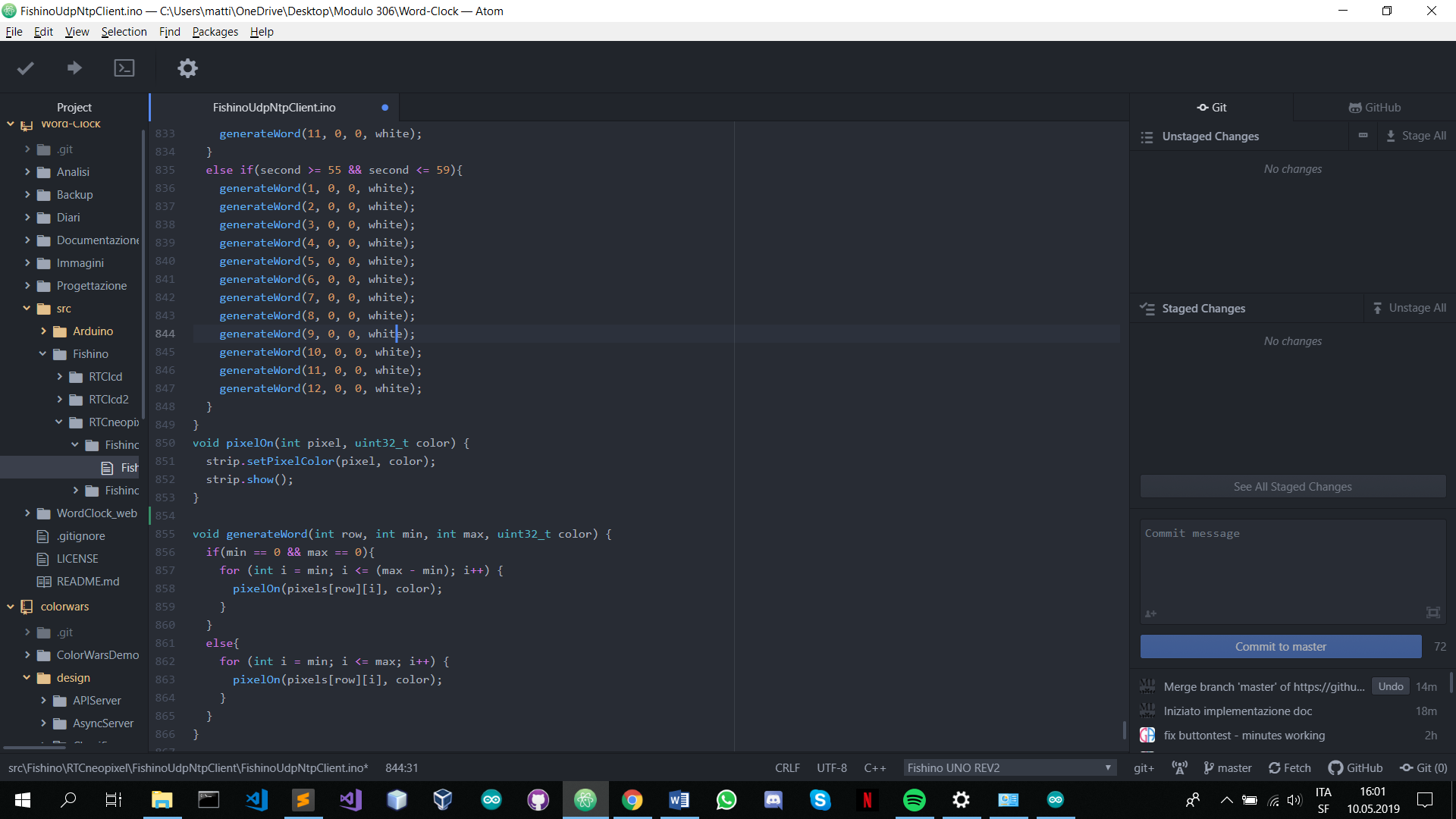


In seguito nel metodo Loop() andiamo a prendere l’orario dall’RTC e inseriamo le ore, i minuti e i secondi in delle variabili apposite.

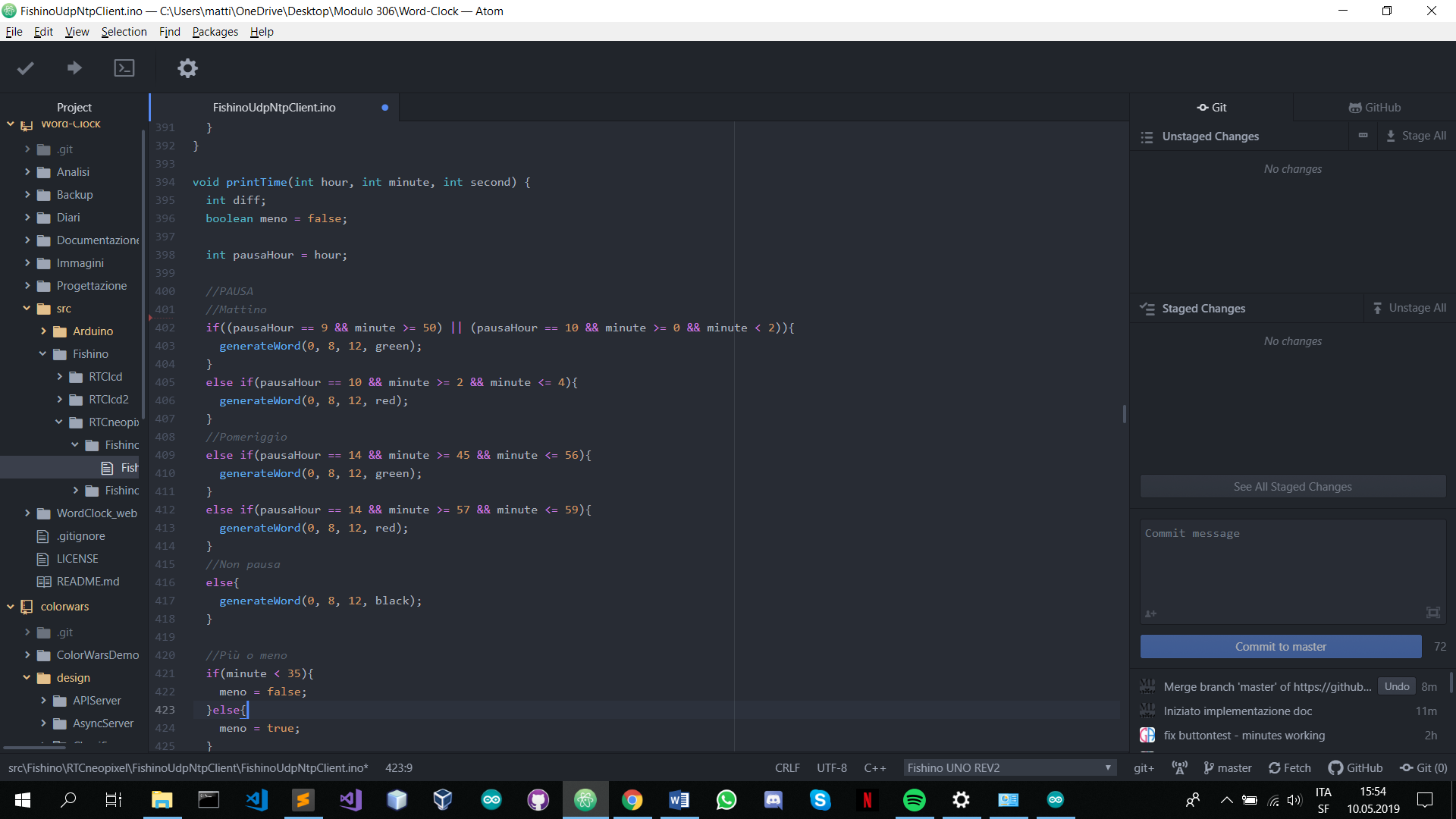


Nella nostra classe principale del progetto abbiamo il metodo printTime(int hour, int minute, int second) in cui svolgiamo praticamente tutte le azioni di accensione dei led. Più tardi sarà possibile trovare la spiegazioni delle parti di codice più importanti presenti in questo metodo, ma per iniziare occorre spiegare il metodo generateWord(int[] word, uint32\_t color).

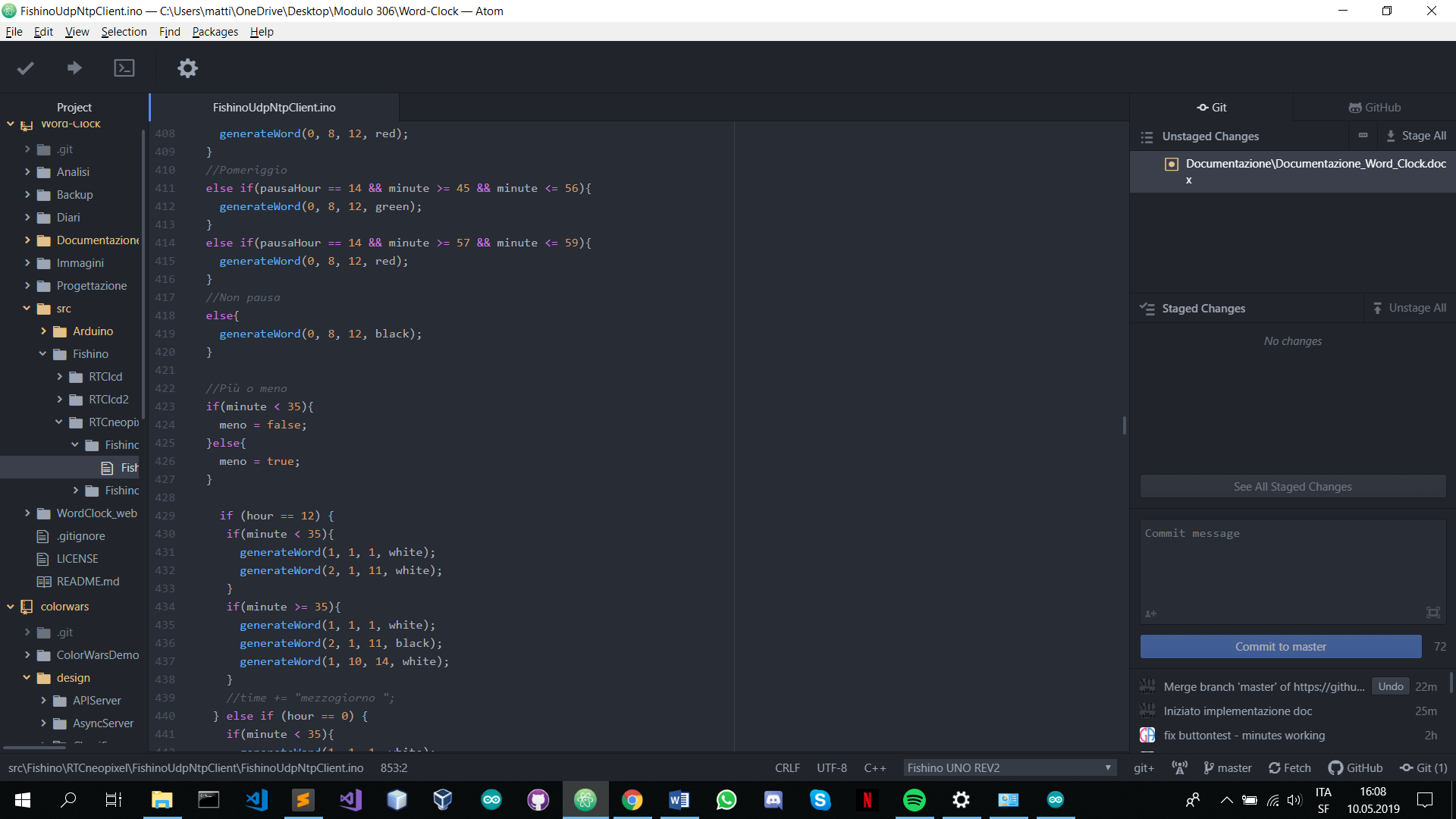
Il seguente metodo, illustrato qua sotto, riceve come primo parametro l’array contenente tutte le informazioni dei led da accendere. Il secondo parametro indica il colore con cui si devono accendere i led.



Passiamo ora al metodo printWord(). Per iniziare troviamo una serie di if per accendere in verde la scritta “pausa” presente sull’orologio. Questa scritta si accende nella mattina dalle 9:50 alle 10:05 e nel pomeriggio dalle 14:45 alle 15:00. Quando mancano meno di 3 minuti alla fine delle pause la scritta passa da verde a rossa. Per tutto il resto del giorno la scritta pausa rimane spenta.



I seguenti if ed else servono per determinare il valore della variabile booleana “meno”. Nel caso che i minuti siano meno di 35 la variabile viene impostata a false perché la scritta “meno” non deve venire accesa nell’orologio dato che si userà “e” (ad esempio: “sono le tre **e** dieci”). Se invece i minuti sono maggiori di 35 la impostiamo a true perché la scritta “meno” deve accendersi (ad esempio: “sono le quattro **meno** dieci”).



# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? Ecc

# Bibliografia

## Bibliografia per libri

* Massimo Del Fedele, *Fishino*, Futura Group srl, 2017, 978-88-909529-5-1.

## Sitografia

* <https://fishino.it/fishino-uno-it.html>, *documentazione fishino uno*,15.02.2019
* <https://atom.io/packages/git-plus>, *auto fetch github*,15.02.2019
* <http://www.fishino.com/caricamento-wireless-degli-sketch-ota.html>, *caricamento wireless degli sketch (ota)*, 22.02.2019
* <https://www.markdowntutorial.com/lesson/4/>, *inserire immagine in MD,* 22.02.2019
* <https://aticleworld.com/http-get-and-post-methods-example-in-c/>, *richiesta GET in C per connessione time server*, 22.02.2019
* <http://www.fishino.com/arduino-ide-packages-it.html>, *packages per l'ide di arduino – fishino*, 22.02.19
* <https://www.adafruit.com/product/2969>, *ricerca informazioni led NeoPixel*, 22.02.2019
* <http://www.fishino.com/download-libraries-it.html>, *librerie – Fishino*, 22.02.2019
* <http://www.fishino.com/download-drivers-it.html>, *drivers – Fishino*, 22.02.2019
* <https://www.wikiinfo.net/t2800-come-funziona-rtc-del-fishino-codice>, come funziona RTC del Fishino, 27.02.2019

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o Qdc
* Prodotto
* …