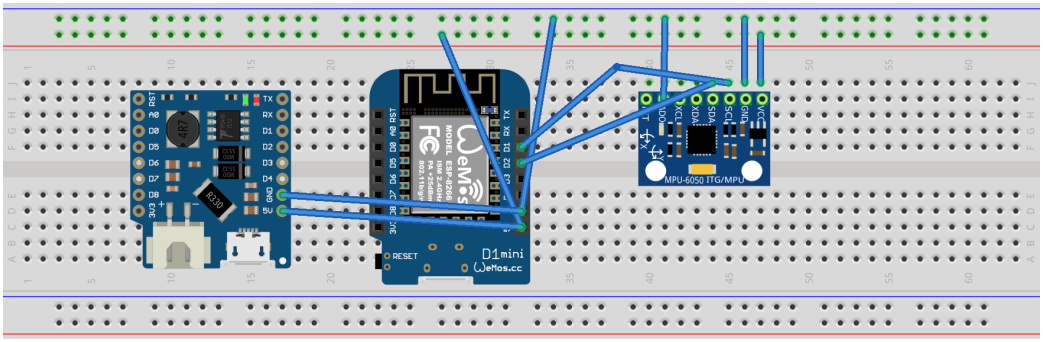
Implementation

1 Hardware

1.1 Circuito

Viene riportato di seguito il circuito che si trova all'interno del giocattolo:



Le componenti del circuito sono:

- Wemos D1 mini
- MPU-6050, accelerometro a 6 assi
- Wemos Battery Shield

1.2 Cablaggio

Battery Shield	D1 mini
5V	5V
GND	G

D1 mini	MPU-6050
5V	VCC
G	GND
D1(SCL)	SCL
D2(SDA)	SDA
G	AD0

1.3 Descrizione

- La componente MPU-6050 è in grado di misurare l'accelerazione della scheda lungo gli assi x,y,z. Tale accelerazione è proporzionale alla somma vettoriale di tutte le forze applicate. Nel progetto il sensore viene utilizzato per fare motion detection, cioè le accelerazioni misurate vengono rielaborate per decidere se la scheda(e il giocattolo di conseguenza) viene spostata lungo una certa direzione: in particolare, se la scheda viene mossa lungo una certa direzione con un certo verso, il sensore misurerà un'accelerazione di una intensità proporzionale alla forza appplicata lungo tali direzione e verso(proiettata lungo gli assi x, y, z su cui avvengono le misurazioni). Dati gli inevitabili errori di misura dello strumento, i dati raccolti e inviati dal circuito verranno sottoposti all'azione filtrante di un filtro passa-basso(vedi sezione software).

- La scheda Wemos D1 mini, oltre a fornire il microcontrollore che esegue lo sketch riportato nella sezione software, viene usato come Access Point creando una rete internet ad-hoc. Il videogioco si connette alla scheda generando una connessione TCP al momento della generazione del percorso. Il D1 mini si comporta quindi da server inviando le misure ottenute dall'accelerometro al videogioco.

- La Wemos Battery Shield consente di alimentare il circuito usando una batteria al litio, di dimensioni ridotte e ricaricabile.

2 Software

La parte software del progetto è divisa in due componenti principali:

- Lo sketch eseguito dal circuito nel giocattolo
- Il videogioco eseguito dal tablet

L'utente interagisce solo con la seconda componente mentre la prima deve rimanere il più possibile trasparente. Queste due parti comunicano tra loro solo durante la fase della registrazione di un percorso su cui l'utente può gareggiare nel videogioco. Seguono la descrizione delle principali caratteristiche e funzionalità per entrambe, insieme a decisioni e dettagli a livello implementativo. Tutto il materiale relativo al progetto può essere trovato al seguente link: **https://github.com/FabioCastle5/wil-auiproject.git**

2.1 Hardware Sketch

Il software eseguito dal circuito presente nel giocattolo risiede in uno sketch salvato ed eseguito dal microcontrollore(vedi sezione hardware). Tale sketch è stato sviluppato usando la IDE ufficiale di Arduino(versione attuale 1.8.5), che è compatibile con la scheda Wemos D1 mini utilizzata.

Caratteristiche

- **Leggerezza:** Lo sketch deve avere delle dimensioni ridotte e richiedere una potenza di calcolo adatta a un microcontrollore.
- **Disponibilità:** Il videogioco deve ricevere delle misure dal circuito ogni volta che ne ha bisogno.
- **Indipendenza:** La comunicazione tra circuito e tablet dev'essere possibile indipendentemente dalla presenza di una rete internet pubblica.

Funzionalità

Le funzionalità principali dello sketch sono la creazione di una rete internet a cui il videogioco si può connettere per ricevere le misure dell'accelerometro e l'invio di tali misure quando richiesto dal videogioco stesso. Affinchè possano essere inviate delle misure significative, lo sketch calcola inoltre il valore medio dell'errore di misura e lo elimina dai dati ricevuti dall'accelerometro.

- **Access Point:** il Wemos D1 mini viene configurato in modalità Access Point. In questo modo il tablet può connettersi attraverso una connessione TCP con la scheda senza che debba essere presente una rete internet pubblica tra i due.

- **Errore medio:** il calcolo dell'errore medio di misura viene fatto quando il circuito è "a riposo", cioè in fase di avviamento dello sketch e quando non c'è alcuna comunicazione con un client. L'assunzione che è stata fatta è che finchè non viene avviata la fase di registrazione del percorso, il giocattolo rimanga in posizione orizzontale su una superficie non inclinata. La formula utilizzata per il calcolo della media è quella riportata nello sketch.

- **Invio delle misure:** La comunicazione delle misure tra scheda e tablet viene avviata e terminata dal videgioco(sul tablet). La scheda si comporta da server che aspetta delle richieste da parte di un client.

Sketch

Vengono riportate le parti principali dello sketch.

```
1 // include neccessary libraries and define variables
     and constants
2
3 void setup() {
4   // initialize the accelerometer and serial
       communication
5
6     // setup serial communication and wifi
7     WiFi.mode(WIFI_AP);    // Access Point
8     WiFi.softAP(ID, PASS);  // Set (SSID, password)
9     server.begin();      // Start the HTTP Server
10
11    // read some initial values and discard them
12
13    // evaluate the mean error
14    evaluate_offset(AVG_SAMPLES);
15 }
16
17 void evaluate_offset(int samples) {
18    for (int i = 0; i < samples; i ++) {
19    // setup avg_samples and aX, aY
```

```
20
21      // incremental mean formula
22      avgx = (aX + (avg_samples - 1) * avgx) /
           avg_samples;
23      avgy = (aY + (avg_samples - 1) * avgy) /
           avg_samples;
24  }
25
26  void loop() {
27      // listen for connecting client
28      client = server.available();
29      if (client) {
30        // there is a client connected <-> the tablet is
              listening
31        while (client.connected()) {
32            // take a measure of acceleration and send
                 data to client
33            measure_and_send();
34        }
35        // client disconnected
36      }
37      // update mean error
38      delay(500);
39  }
40
41  void measure_and_send() {
42      // measure aX, aY from accelerometer
43
44    // discard the mean error
45      aX = aX - avgx;
46      aY = aY - avgy;
47
48      //send measure to client - format: (ax);(ay)\r
49  }
```

Nota Come si può osservare, lo sketch non prevede il filtraggio delle misure dell'accelerometro. Tale funzionalità è affidata al videogioco, insieme a ulteriori trasformazioni ai dati inviati dalla scheda che consentano di costruire un percorso giocabile.

2.2 Videogioco

Il videogioco è stato sviluppato utilizzando la IDE Unity3D. Gli script inseriti nel progetto e riportati nel documento sono scritti nel linguaggio CSharp. Come ogni progetto sviluppato in Unity3D, questo è suddiviso in scenes che sono dei blocchi funzionali, ciascuno con le proprie funzionalità e contenente i propri oggetti e script.

Struttura

Il progetto è suddiviso in 4 scenes.

Welcome scene

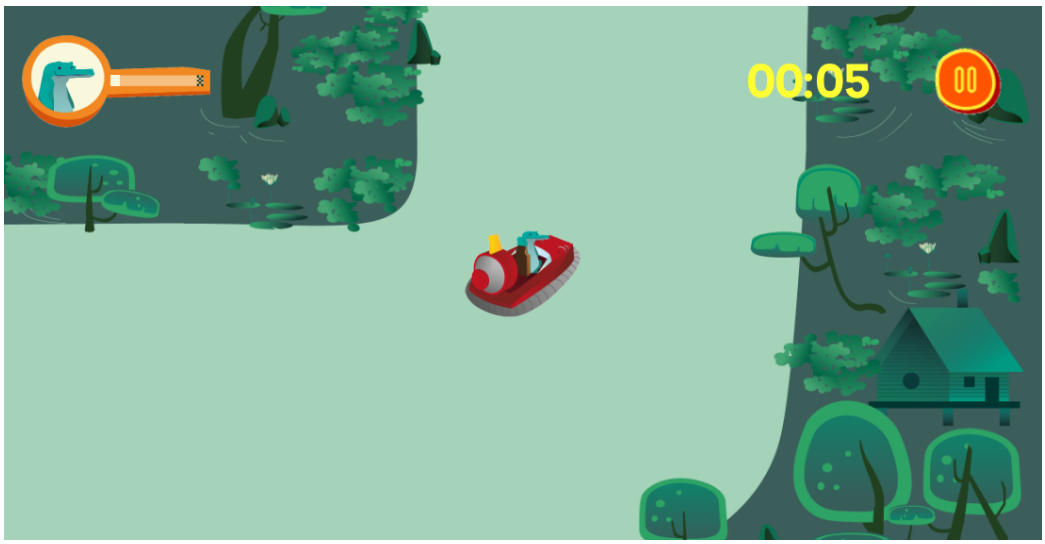
- Consente di scegliere tra la Game scene e la View Circuit scene
- Avvia la registrazione del circuito
- Consente di scegliere una lunghezza media del percorso che verrà costruito

Circuit Registration scene

- Avvia la comunicazione tra tablet e giocattolo e la ricezione delle misure inviate da quest'ultimo
- Rielabora e filtra i dati ricevuti
- Avvia una delle due scenes finali

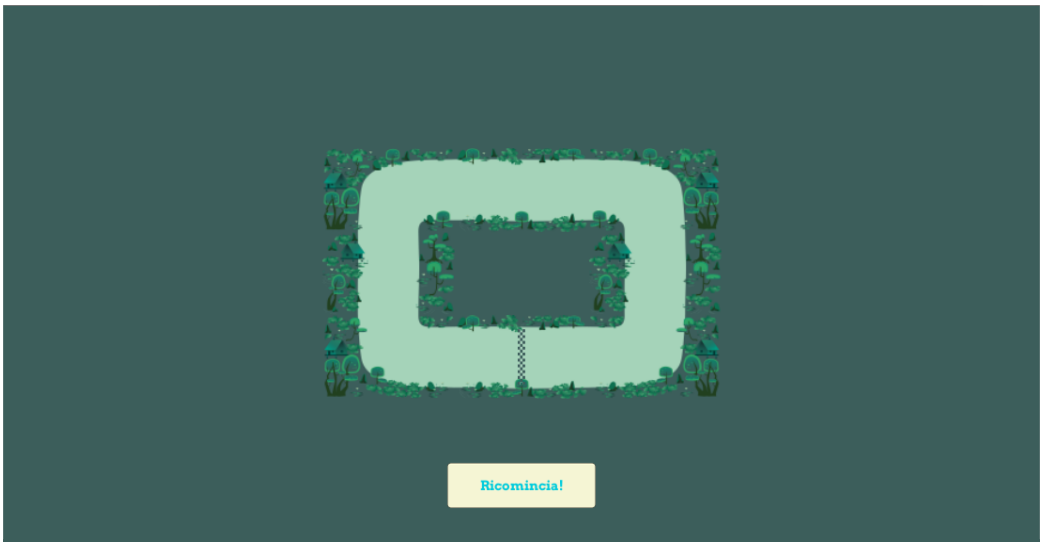
Game scene

- Costruisce il circuito digitale
- Avvia e gestisce la fase di gioco



View Circuit scene

- Costruisce il circuito digitale
- Consente di visualizzare il circuito appena costruito nella sua interezza



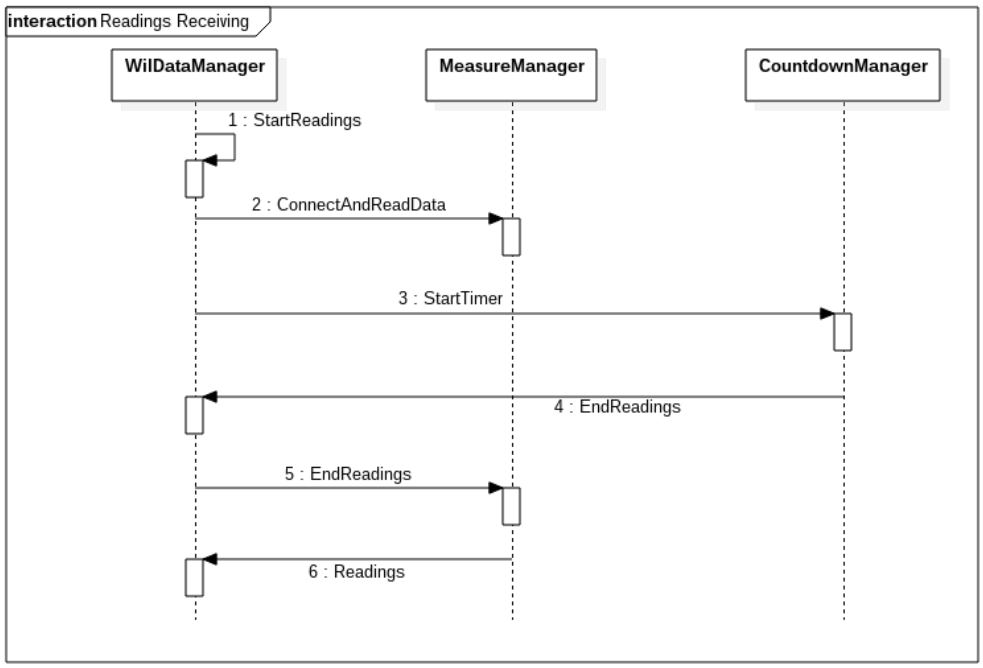
Ricezione delle misure

Questa funzionalità viene gestita nella seconda scene (Circuit Registration) e sono tre le principali classi che vengono interpellate:

- **WilDataManager:** Interagisce con la GUI e avvia sia la connessione con il giocattolo(e la successiva ricezione delle misure da parte della scheda che contiene) che il timeout.

- **CountdownManager:** Controlla il timer e notifica la classe principale(WilDataManager) quando la finestra temporale di ricezione delle misure da parte del giocattolo si è esaurita.

- **MeasureManager:** Esegue la connessione TCP con la scheda e legge e salva le misurazioni che vengono inviate da quest'ultima; quando il tempo di ricezione delle misure è scaduto termina la connessione e invia le misure ricevute alla classe principale(WilDataManager)



filtraggio e rielaborazione dei dati

Una volta ricevute le misure da parte del giocattolo è necessario rielaborarle affinchè si possa ricostruire il percorso compiuto fisicamente a livello digitale. Ricevute le varie accelerazioni effettuate delle scheda è dunque necessario decifrare come la scheda si è spostata nello spazio. Tutte le trasformazioni necessarie vengono eseguite nella classe *WilDataManager* nel metodo *ElaborateEntries()*, una volta terminata la comunicazione con il giocattolo. I dati vengono inizialmente filtrati, dopodichè viene eseguita una trasformazione delle misure di accelerazione in motion detection.

Filtraggio Per raffinare i dati ricevuti e renderli meno soggetti ad errore, questi vengono filtrati attraverso un filtro passa-basso. Il filtro legge tutte le misure ricevute e, basandosi sulla misura in esame e quella precedente, modifica la misura secondo la relazione:

$$m_i = factor * m_i + (1 - factor) * m_{i-1}, factor \in [0, 1]$$

factor è un valore ottenuto sperimentalmente confrontando l'effetto di alcuni suoi valori sulla precisione della misura effettuata. Un valore vicino allo 0 tende ad "appiattire" tutte le misure di accelerazione ottenute sull'asse 0,

mentre un valore di factor pari a 1 non induce alcuna modifica sulle misure ricevute.

Motion Detection Viene definita una certa *soglia* che separa i dati interessanti da quelli trascurabili. Se una misura di accelerazione è inferiore a tale soglia(in valore assoluto), allora tale misura è trascurabile, nel senso che è dovuta a un errore di misurazione o che è dovuta a uno spostamento trascurabile del giocattolo. Il valore della soglia influenza la sensibilità allo spostamento della scheda. Confrontare le misure di accelerazione lungo un certo asse(x,y) con la soglia consente di stabilire se tale misura corrisponde a uno spostamento del giocattolo lungo tale asse(+1 o -1) o se il giocattolo è rimasto fermo(0). Il movimento del giocattolo viene quindi approssimato come una sequenza di spostamenti, ciascuno lungo 8 possibili direzioni(N, NE, E, SE, S, SW, W, NW).
Sia threshold il valore della soglia, siano xList e yList liste di valori che rappresentano le misure di accelerazione lungo gli assi x e y(il cui valore è confrontabile con quello di threshold) e siano moveX e moveY le relative liste di dati di motion detection:

```
int movex, movey;

// convert data into motion detection values
for (i = 0; i < xList.Count; i++) {
        // movex
        if (xList[i] > threshold) {
                movex = 1;
        } else if (xList[i] < -threshold) {
                movex = -1;
        } else {
                movex = 0;
        }
        //movey
        if (yList[i] > threshold) {
                movey = 1;
        } else if (yList[i] < -threshold) {
                movey = -1;
        } else {
                movey = 0;
        }
        moveX.Add(movex);
        moveY.Add(movey);
```

```
}
```

Ulteriori rielborazioni Affinchè sia possibile costruire il percorso a partire dai nuovi dati ottenuti nelle liste moveX e moveY, questi devono essere rielaborati. In particolare:

- Vengono eliminati tutti i punti che non provocano alcuna modifica nella costruzione del percorso(gli spostamenti 0,0). Questi punti sono i valori j per cui moveX[j] e moveY[j] sono entrambi nulli.
- Vengono eliminati tutti gli spostamenti che sono opposti a quelli precedenti. Questi spostamenti indurrebbero il percorso a "ritornare sui propri passi", che è un comportamento difficile da gestire. Spesso questo comportamento è conseguenza di un errore di misura. Per ogni spostamento (moveX[j], moveY[j]) vengono eliminati i successivi spostamenti ruotati di 180 gradi rispetto al j-esimo,cioè tutti i k per cui moveX[j] + moveX[k] = 0 e moveY[j] + moveY[k] = 0.

Posizione Una volta raffinati tutti gli spostamenti nelle liste moveX e moveY, vengono calcolate le liste posX e posY che rappresentano la posizione nello spazio del giocattolo in seguito ad ogni spostamento. Il percorso inizia sempre in posizione (0,0). Nel caso in cui uno spostamento imponesse una posizione in un punto già raggiunto in precedenza(il percorso avrebbe degli overlapping), tale spostamento viene ignorato.

```
posX.Add (0);
posY.Add (0);
int lastx = 0;
int lasty = 0;
int px = 0;
int py = 0;

for (i = 0; i < moveX.Count; i++) {
        px = lastx + moveX [i];
        py = lasty + moveY [i];
        if (!PointAlreadyPresent(posX, posY, px, py)) {
                // no overlapping point
                posX.Add (px);
                posY.Add (py);
                lastx = px;
                lasty = py;
```

```
            }
}
```

Il grafico ottenuto da (posX, posY) è l'approssimazione del movimento del giocattolo con cui verrà costruito il percorso digitale.

Creazione del percorso

La prima fase della scene *Game* è quello di costruire il circuito in cui far giocare l'utente. Questa funzionalità è gestita dalla classe *DataManager*. Partendo dalla lista di posizione calcolata nella fase di registrazione del percorso, viene generato il circuito vero e proprio generando una lista di tiles predefinite che posizionate una dopo l'altra ricalchino il percorso registrato. Le tiles gestiscono gli spostamenti nelle 8 direzioni e in più sono previste delle curve a 90 gradi affinchè ciascuna possa essere collegata alle altre. L'algoritmo ad alto livello che viene seguito è il seguente:

1. Viene disegnata la tile iniziale che comprende la start flag;
2. Per ogni punto presente nella lista di posizione viene valutato se è necessario disegnare una tile di curva per collegare la tile precedente a quella successiva, dopodichè viene disegnata la tile successiva;
3. Viene disegnata la tile finale che comprende la finish flag.

Di seguito è riportato un esempio di circuito costruito a partire da una lista di punti:

X	Y
0	0
1	0
1	1
0	1
0	0
-1	-1
-2	0
-2	1