# Introduzione alla Programmazione e al Physical Computing Edutainment con micro:bit

Questo documento propone lo sviluppo di una serie di giochi interattivi a complessità crescente, dal software puro al physical computing, con l’obiettivo di fare scoprire agli studenti il più possibile autonomamente i concetti fondamentali del coding, ma soprattutto acquisire computational thinking skills utili in un contesto più ampio, con un approccio costruttivista. Consente quindi di acquisire competenze informatiche fondamentali in un contesto laboratoriale ludico e coinvolgente, utilizzando tecniche di programmazione ad apprendimento rapido (programmazione visuale) e metodologie didattiche fondate sulla sperimentazione, la makers education e la scoperta (discovery learning).

## Indicazioni pedagogiche

Il docente deve proporre la sequenza di progetti in modo molto flessibile, apportando le variazioni necessarie per valorizzare opportunisticamente le “*scoperte*” degli studenti e i loro suggerimenti, che non sono completamente anticipabili.

Il docente, con scarne spiegazioni preliminari, suggerimenti mirati e il supporto eventualmente necessario, deve mantenere le varie proposte a un livello di difficoltà tale da *stimolare curiosità e motivazione*, evitando tuttavia che gli studenti si sentano frustrati perché incapaci di avanzare. Questo richiede un monitoraggio attento e un supporto fortemente individualizzato. Una tecnica che si è rivelata ottima è proporre l’esercizio su una learning platform e chiedere agli studenti di condividere fin da subito il link al loro lavoro sull’emulatore. In questo modo il docente può seguire i vari studenti in tempo reale evitando tuttavia una presenza fisica a volte ingombrante, per identificare chi ha bisogno di supporto, chi può trarre vantaggio da qualche suggerimento mirato, chi ha bisogno di stimoli ulteriori, ed intervenire quindi nel modo più appropriato. Questo permette inoltre di identificare opportunità per discutere con tutti gli studenti eventuali soluzioni parziali interessanti, o sollecitare/proporre suggerimenti per superare ostacoli imprevisti.

È importante chiedere agli studenti di privilegiare sempre la soluzione più semplice possibile (*simple is beautiful*). Utile organizzare contests amichevoli dove vince chi propone la soluzione più semplice che soddisfa le specifiche proposte. In questo caso è importante che siano gli studenti a decretare il vincitore, in modo che sviluppino *skills metacognitivi* che favoriscano le loro capacità di progredire in modo autonomo.

Questo documento fornisce inizialmente istruzioni dettagliate con esempi completi di soluzioni, per poi lasciare sempre più spazio alla iniziativa degli studenti e dei docenti. Il documento è stato sviluppato principalmente come materiale di supporto ai docenti, ma può essere convenientemente utilizzato anche come materiale di supporto per gli studenti, avendo cura di somministrarlo in parti progressive senza mai fornire soluzioni anticipate.

## Materiale di riferimento

Presentazione del dispositivo [http://micro:bit.org/guide/features/](http://microbit.org/guide/features/)

Ambiente di sviluppo con simulatore [https://makecode.micro:bit.org/](https://makecode.microbit.org/)

Pins description [Edge Connector and Pinout](https://tech.microbit.org/hardware/edgeconnector/)

Simulatore circuitale con micro:bit <https://www.tinkercad.com/things/0O5Lksq72nO-microbit>

## Overview della struttura fisica e funzionale del micro:bit

Presentazione della scheda fisica, in particolare:

* Microcontrollore: integra CPU, memoria volatile, memoria permanente e vari dispositivi di I/O
* dispositivi di output: matrice di led, loudspeaker
* dispositivi di input: pulsanti, magnetometro, accelerometro, sensore di luce, sensore di temperatura, microfono.

<http://micro:bit.org/guide/features/> <https://microbit.org/get-started/features/overview/>

Cenno alla possibilità di interagire con il mondo fisico tramite ulteriori sensori e attuatori esterni (sensore di umidità, servomotori, etc.) collegabili tramite i contatti esterni (edge connector).

Presentazione dell’ambiente di sviluppo con simulatore, con le principali istruzioni di input output: [https://makecode.micro:bit.org/](https://makecode.microbit.org/)



## Testa o croce [algoritmo, variabile, selezione: IF-THEN-ELSE, evento]

Primo esempio da dimostrare, di cui chiedere poi lo sviluppo personalizzato in autonomia

Task: sviluppare un programma per simulare il lancio di una moneta.

Algoritmo:

Se un valore logico (true or false) generato “casualmente” è true, mostra croce, altrimenti testa.

QUANDO viene scosso il dispositivo:

genera un valore random true o false

IF (valore generato random è true)

THEN

mostra croce

ELSE

mostra testa

ENDIF

Nota: (valore generato random è true) <=> (valore generato)

Prerequisiti:

* Show leds: mostrare immagini di esempio
* Mostrare comportamento di On start / On shake: show string “Hi”

|  |  |  |  |
| --- | --- | --- | --- |
| * If then else: utilizzare if con costante (IF TRUE) e mostrare cosa avviene cambiando la costante (IF FALSE): | |  | |
| * Utilizzare una variabile logica: | |  | |
| * Pick random true or false: utilizzare al posto della costante precedente (con evento “shake”) | |  | |
| * Codifiche alternative con impiego di variabile numerica (più facilmente utilizzabile nell’esercizio successivo): |  | |  |

Challenge: moneta truccata - fare in modo che croce esca con una probabilità superiore al 50/100

## Dado 1..6 [IF-THEN-ELSE multipli, in sequenza e in nesting]

Task: Simulare il lancio di un dado.

|  |  |
| --- | --- |
| Versione base con show number: |  |

#### Challenges

|  |  |
| --- | --- |
| * Visualizzare le facce del dado.   Prerequisito: IF-THEN – mostrare esempio se necessario: |  |

* Mostrare soluzione più efficiente con ELIF
* Modellare un dado truccato
* Modellare due dadi o la somma di due dadi
* verificare la frequenza di occorrenza di vari eventi (con due dadi: almeno un 6, due 6, due volte due 6 consecutivi…)

## Beating heart [ripetizione – loop infinito]

*Algoritmo*:

FOREVER:

BEGIN

Mostra cuore grande

Aspetta 1 secondo

Mostra cuore piccolo

Aspetta ½ secondo

END

Prerequisito: istruzione pause

|  |  |
| --- | --- |
| Codifica in Block language: |  |

## Beating heart - estensione [Gestione eventi – pulsanti A e B]

Utilizzando l’esempio seguente, spiegare brevemente:

* Concetto di gestione eventi (ON button A pressed...)
* Concetto di variabile condivisa controllata dagli eventi



Tasks:

* Fare in modo che il cuore pulsi sin dall’inizio
* Aumentare o diminuire il ritmo cardiaco del cuore precedente premendo pulsanti A o B
* Modellare arresto cardiaco con A e B contemporaneamente premuti
* Modellare la rianimazione con l'evento shake

Challenges:

* Fare in modo che possa essere rianimato solo entro dieci secondi dall’arresto cardiaco, altrimenti sopravviene la morte.
* Modellare aritmia (un battito assente random, con probabilità 1/5)

|  |  |
| --- | --- |
| Esempio soluzione base: |  |

## Rock-Paper-Scissor game parziale [IF-THEN-ELSE in sequenza e nidificati]

Task: Sviluppare algoritmo in pseudocodice (attenzione alla indentazione corretta) e programmare il micro:bit affinchè faccia una singola giocata Sasso/Forbici/Carta:

* Versione con IF in sequenza (IF THEN ENDIF IF THEN ENDIF…)
* Versione con IF nidificati (IF THEN ELSE IF… ENDIF ENDIF)
* Confrontare l’efficienza delle due soluzioni
* Scrivere lo pseudocodice corrispondente ai due casi



## Testa o croce con rotolamento della moneta (estensione difficile ora, ma qualche studente dovrebbe riuscirci) [Ciclo condizionato]

Task: Visualizzare prima del risultato finale una rapida sequenza di testa/croce

#### Challenges

* rallentare progressivamente la velocità della sequenza   
  hint: [while (delay < 3000) show…, increase delay, show…, increase delay]

## Scorekeeper gadget [espressioni aritmetiche, user interface]

Task: Sviluppare un gadget per tenere traccia del punteggio di due giocatori (il gadget sarà riutilizzato, in forme anche differenti, negli esercizi successivi):

* Il pulsante A incrementa il punteggio del giocatore1, B del giocatore2. Attenzione alla user interface: come far capire all'utente che ha schiacciato il pulsante?
* Piegando il micro:bit a sinistra visualizza il punteggio di A, a destra quello di B.
* A+B resetta il conteggio (riparte da 0).

|  |  |
| --- | --- |
| Inizializzazione: |  |
| Avanzamento punteggio: |  |
| Visualizzazione punteggi attuali: |  |

#### Challenges

* Gestire il controllo di vincita partita (per esempio il gadget annuncia quando uno dei due punteggi vale 5).
* Gestire partite di tre set. Quando un giocatore arriva a 10 micro:bit decreta la vincita del set e si ricomincia un nuovo set. Quando un giocatore vince tre set, vince la partita.
* Visualizzare i punteggi con una fila di led accesi, la cui altezza indica il punteggio nella partita corrente. Essendoci solo cinque led per colonna, come fare per gestire i 10 punti? Proporre due soluzioni differenti. [hint: si potrebbero utilizzare le istruzioni plot x y...]

## Gioco interattivo sasso/carta/forbici: user contro micro:bit con scoring [eventi]

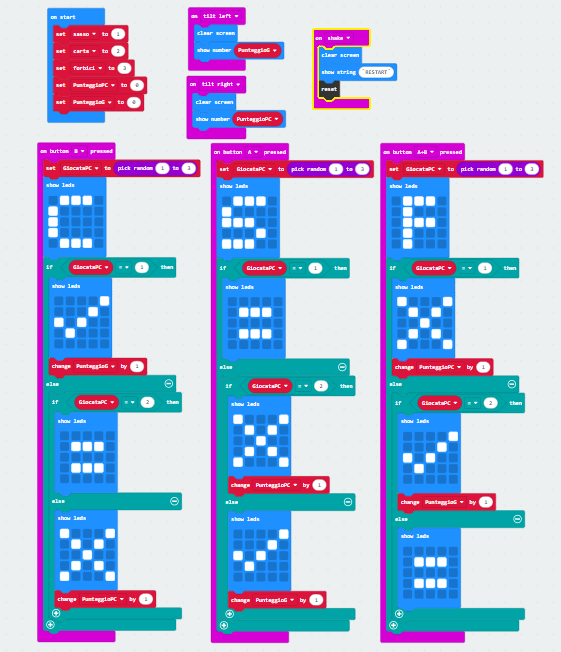
Tasks: Sviluppare il gioco sasso/carta/forbici completo:

* Il giocatore gioca la propria mossa (A/B/A+B);
* Il micro:bit visualizza la mossa del giocatore, genera e visualizza la propria mossa random, visualizza YOU LOSE/WIN; aggiorna e visualizza il punteggio.
* Il giocatore può accedere alle istruzioni per il gioco.
* Tilt left/right mostrano il punteggio attuale di micro:bit/giocatore

Progettare l’applicazione in pseudocodice (attenzione alla indentazione):

* Attenzione al nome delle variabili (deve riflettere chiaramente la loro semantica). Può essere utile avere Sasso=1, Carta=2..., UserScore, ComputerScore...
* Produrre la documentazione opportuna (analisi, progetto, codifica, testing…).

Esempio di soluzione:



#### Challenges

* Controllare il raggiungimento di fine partita.
* Fare in modo che il micro:bit bari vincendo più spesso (20%) del normale.

## Input number gadget [user interface]

Sviluppare un gadget (che verrà utilizzato in seguito) per introdurre un numero con i pulsanti A e B: A +10 (incremento decine), B +1 (incremento unità). Il pulsante A+B termina (conferma) l’operazione (rendendo impossibile modificare ulteriormente il numero introdotto).

#### Challenges

* reset on shake
* consentire il conteggio decine/unità fino a 9 e ripartire da zero.
* toggle il verso (up/down) del conteggio con shake
* disabilitare la lettura con timeout
* aumentare/diminuire con tilting left/right

## HILO – vanilla version [cicli]

Task: Il micro:bit visualizza un numero random fra 0 e 100. Chiede poi al giocatore di indovinare se il prossimo numero generato sarà maggiore o minore. Se il giocatore indovina guadagna un punto. Il gioco termina quando il giocatore non indovina.

Produrre pseudocodice opportunamente commentato, e codice opportunamente testato.

#### Challenges

Modificare il programma precedente, aumentando la probabilità di perdita da parte del giocatore (senza esagerare). Questo può essere ottenuto, per esempio, rigenerando un nuovo numero random ogni volta che il micro:bit si accorge (segretamente) che il giocatore vincerebbe la partita.

## HILO parziale – standard version [cicli, integrazione]

Task: Sviluppare uno step del gioco interattivo HILO (pseudocodice e codice):

* Generare un numero casuale segreto fra 1 e 100;
* Leggere dall'utente un tentativo di indovinarlo, consentendo le operazioni +1/+10 e

-1/-10 con i pulsanti A e B, e A+B per cambiare il segno (integrazione gadget precedente);

* Scrivere "indovinato" o "alto" o “basso”.
* Esplicitare la strategia di test.

Task: Sviluppare lo pseudocodice e il codice micro:bit per completare il gioco HILO:

* Il micro:bit deve concedere all'utente un massimo di 5 tentativi per indovinare il numero, dopodiché, se l'utente non ha indovinato, dichiara la propria vincita.
* I pulsanti devono venire disabilitati in caso di vincita, e riabilitati con un evento shake per ricominciare il gioco.

#### Challenges

* Consentire all'utente di inserire il numero massimo. Il computer in questo caso deve calcolare anche il numero (ragionevole) di tentativi da concedere all'utente.
* Consentire all'utente di scegliere il livello di difficoltà.

## Scorrimento leds [cicli predeterminati FOR]

Presentare le istruzioni da utilizzare: plot(x,y), unplot (x,y)

Tasks:

|  |  |
| --- | --- |
| * Fare scorrere un led da sx a dx sulla stessa riga |  |
| * Fare scorrere un led da sx a dx sulla stessa riga, poi sulla successiva (for in sequenza) |  |
| * Fare scorrere un led da sx a dx sulla stessa riga, poi iterativamente sulla successiva (for in nesting) |  |

* Fare scorrere un led dall’alto verso il basso sulla stessa colonna, invertire la direzione
* Fare scorrere un led in diagonale.
* Fare scorrere un led verso il basso (alto) a velocità crescente (decrescente)
* Fare scorrere un led con fluttuazioni di velocità casuali verso il basso (fiocco di neve)

## Timer (e cronometro)

Task: sviluppare un timer con funzionalità di upcount, downcount, stop.

Esempi:

|  |  |
| --- | --- |
| Upcount da 0 a 4: |  |
| Downcount da 4 a 0: |  |
| Generalizzare: |  |
| Temporizzare: |  |
| Arrestare quando si preme il tasto B: |  |

#### Challenges

* Resettare quando si premono A e B contemporaneamente
* Modellare un cronometro con start / stop / pause / continue / tempo intermedio / reset

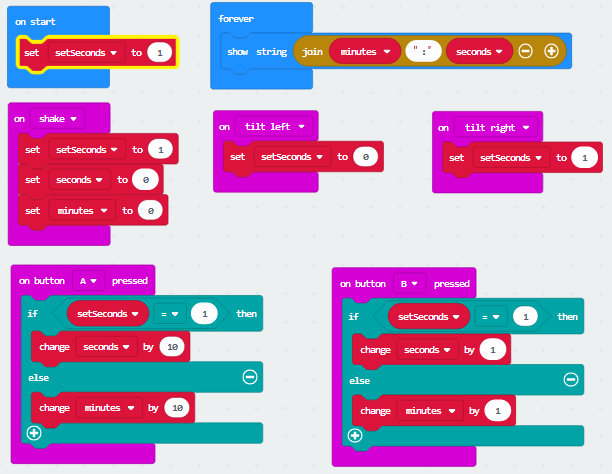
Hints:

* Più funzioni associate al medesimo pulsante a seconda del contesto:
  + A: start se running = FALSE; stop se running = TRUE
  + B: pause…; continue
* Reset: shake
* A+B: tempo intermedio

## Input number gadget (2): timerSetting [user interface]

Task: sviluppare un gadget che consenta l’impostazione di minuti e secondi di un timer [file: micro:bit-timerSetting.hex]:

* Visualizza minuti e secondi nel formato minuti:secondi
* Inizialmente viene settato a 0:0
* Per modificare i minuti si piega a sinistra
* Per modificare i secondi si piega a destra
* Se si preme A aumenta di 10
* Se si preme B aumenta di 1
* Reset by shaking



Warning: il simulatore non funziona perfettamente, dato che per premere il pulsante A si piega anche a sinistra, per premere il pulsante B si piega anche a destra.

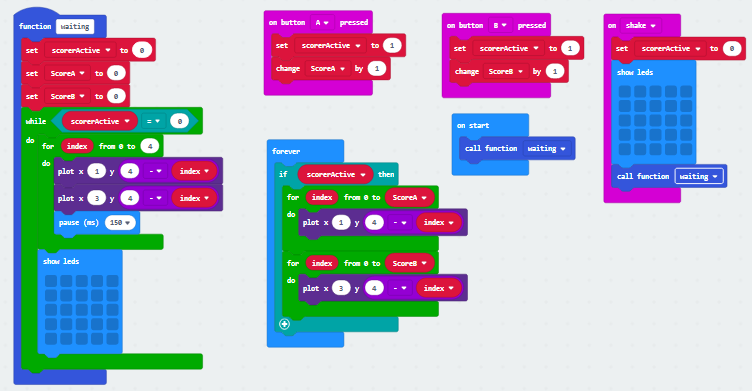
#### Challenges

* Fare in modo che arrivati a 59 il setting riparta da 0 (sia secondi che minuti)
* Fare in modo che A+B inizi il countdown, segnalando il raggiungimento di 0:0
* Utilizzare il formato mm:ss (per esempio 03:06)
* Progettare e sviluppare un’interfaccia alternativa che si ritiene migliore

## Visual ScoreKeeper

Sviluppare uno score-keeper (segna-punti) visuale con le seguenti caratteristiche:

* Quando si preme un pulsante (A o B) si incrementa il punteggio per il giocatore corrispondente (A o B).
* Il punteggio di ciascun giocatore viene visualizzato con una colonna di leds accesi, di altezza proporzionale al punteggio.
* Lo shaking del dispositivo comporta il reset del punteggio.



File: micro:bit-VisualScoreKeeper.hex

#### Challenges

* Perfezionare la soluzione di cui sopra (parte sempre da 1!)
* Quando il punteggio massimo (5) viene raggiunto, un messaggio dichiara il vincitore.
* Modificare la modalità di visualizzazione del punteggio di ciascun giocatore:
  + Il led corrispondente al punteggio raggiunto deve rimanere sempre acceso.
  + La colonna di led sottostanti deve “salire” ripetitivamente da 0 fino al punteggio raggiunto, come se tentasse di spingere il led sempre acceso verso l’alto.
* Ad ogni set vinto, la colonna del giocatore vincente si sposta di una posizione verso il centro (A verso dx, B verso sx).
* Chi vince i primi tre sets vince la partita, sancito da un messaggio e il lampeggio delle tre colonne del vincitore.

Esempio: A ha vinto 1 set, e ha 3 punti nel set corrente. B ha vinto 0 sets, e ha 4 punti nel set corrente:

A B

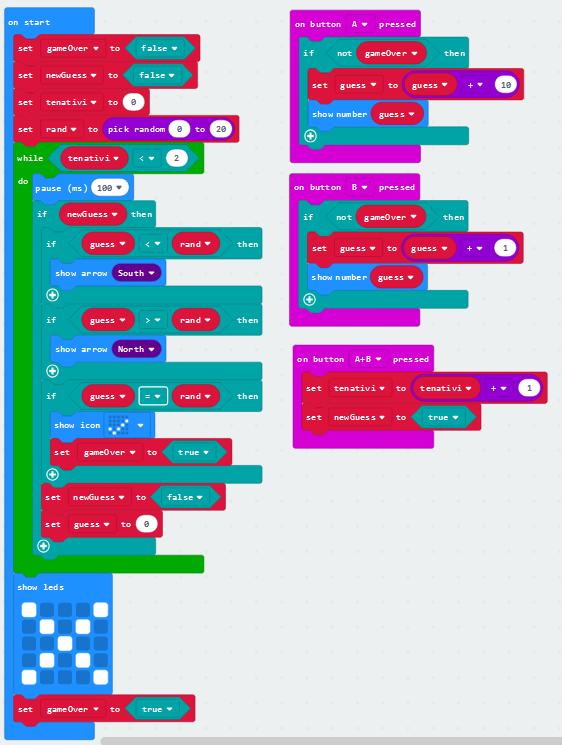
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  | X |
|  | X |  |  | X |
|  | X |  |  | X |
|  | X |  |  | X |

Suggerire altri miglioramenti allo score-keeper.

## HILO completo [ciclo condizionato while]

Utilizzare:

* while (tentativi < 5) nell’evento onStart
* inserire pause (100) per poter rilasciare il controllo ai pulsanti



[file HILO-while.hex]

* Verificare cosa succede senza pause(100)
* Migliorare il codice (terminare il ciclo con AND...) ed estendere funzionalità (nuova partita...)
* Scrivere lo pseudocodice corrispondente, opportunamente commentato

## Trading simulation [cicli, user interface]

Task: Un trader effettua operazioni di compravendita in borsa, guadagnando il 50% su quelle in positivo e perdendo (solo) il 40% su quelle in negativo. Le operazioni in positivo e in negativo sono equiprobabili, e si susseguono casualmente. Quindi il trader dovrebbe, apparentemente, guadagnare... Supponendo che il trader investa inizialmente 10 000 euro, calcolare l’ammontare rimasto dopo 1000 operazioni.

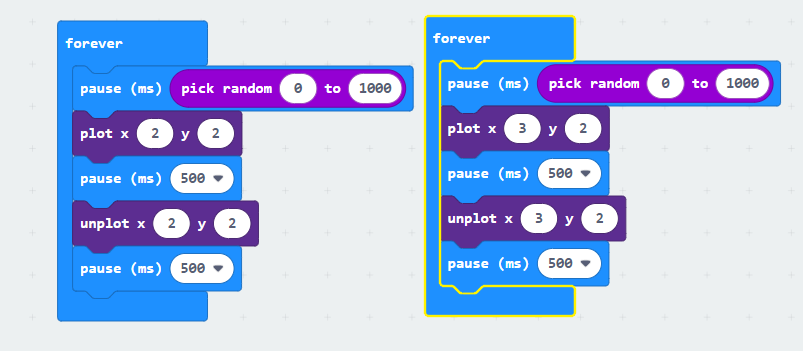
#### Challenge

Visualizzare in real-time l’andamento del valore totale del suo portafoglio.

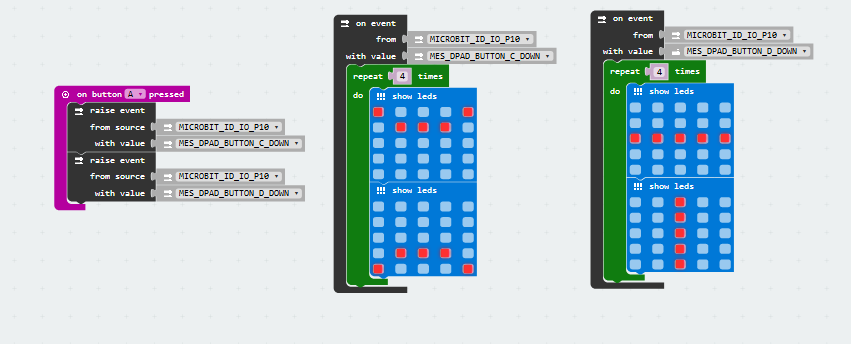
## SlotMachine [concorrenza]

Task: Sviluppare una slot machine stile classico con un led che “scorre” su tre colonne a velocità progressivamente minore sino all’arresto: il giocatore vince quando i tre leds sono sulla stessa riga.

Gestione dei “threads” concorrenti:



Oppure:

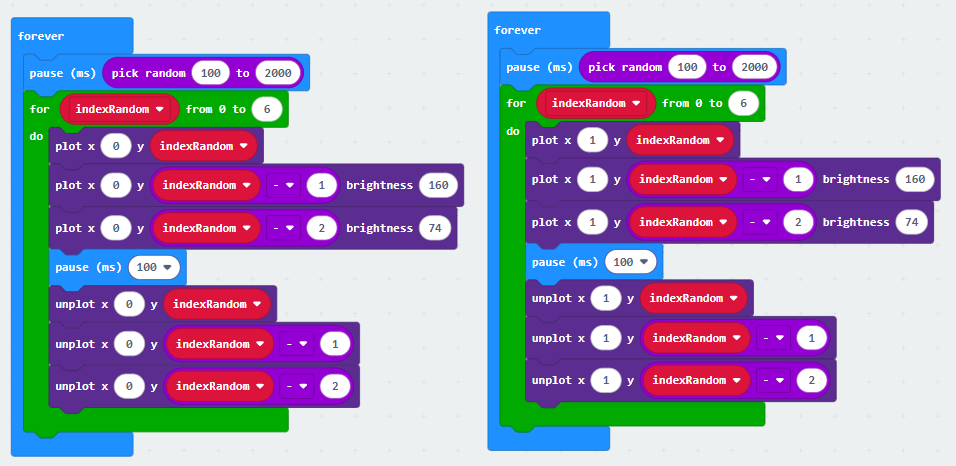


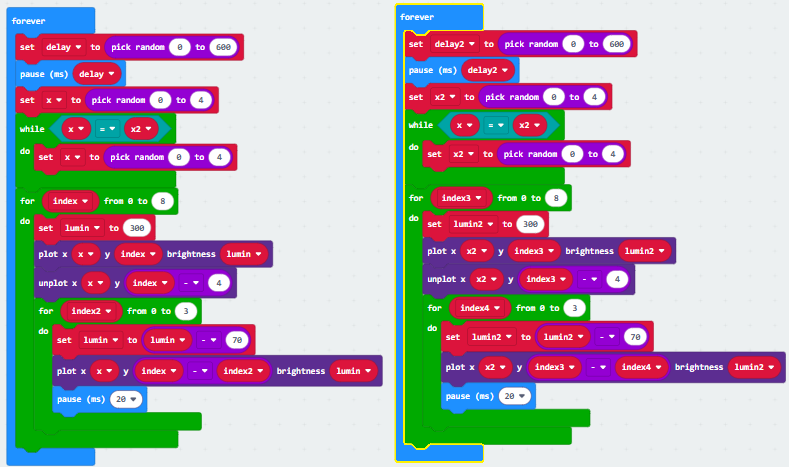
## Shooting Stars (luminarie di Natale) [concorrenza]

Task: Sviluppare il codice per simulare l’effetto di stelle cadenti.

Hint: utilizzare più costrutti forever, con ritardi random (oppure “run in background” con loop “while true”).

Esempi [file ShootingStars1.hex, ShootingStars1.hex]:





## ESERCIZI GENERICI DI SUPPORTO (se necessari)

### REPEAT e FOR

* Sviluppare un programma che visualizza cinque volte “ciao” utilizzando il costrutto FOR.
* Sviluppare lo stesso programma utilizzando il costrutto REPEAT.

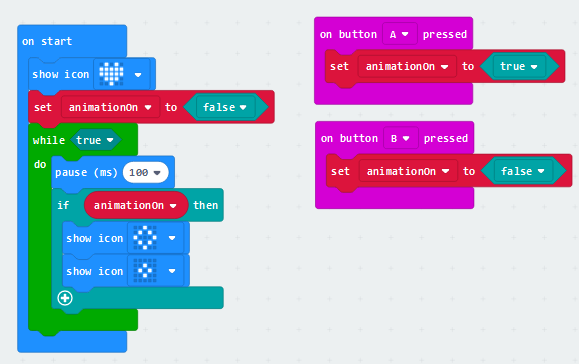
### FOREVER ⬄ WHILE TRUE (Java style) - Two-frames animation

Nota: la variabile di controllo è settata esternamente al loop



### Implementazione equivalente più simile a Java

WHILE: body eseguito “mentre” la condizione è TRUE (notare la pause – altrimenti non rilascia il controllo e si blocca):



* Verificare sperimentalmente cosa succede senza pause(100)
* Breve analisi (funzionalità, ingressi, uscite, variabili)
* Pseudocodice correttamente indentato

### Esercizi numerici

* Calcolare la somma di tutti gli interi compresi fra 1 e un numero dato (fisso)
* Divide et impera (decomposizione funzionale): calcolare la somma di tutti gli interi compresi fra 1 e un numero letto dall’utente (con il meccanismo visto in precedenza A +10 / B +1 / A+B toggles +/-)
* Calcolare la media dei primi 15 numeri pari
* Sviluppare un programmino per scrivere tutti i numeri binari, da 1000 a 1111. Ovvero:

1000

1001

1010

1011

1100

...

Il programmino deve essere facilmente generalizzabile, per esempio per scrivere tutti i numeri da 1000 a 1555 in base 6 (Ovvero: 1000 1001 1002 … 1005 1010 1011 1012 ...)

### WHILE: divisione intera

Calcolare la divisione intera di 437 per 7.

L’idea: 23 DIV 7: 7 14 21 28 STOP (!28<=23) 🡪 3 con avanzo di 2

Lo skeleton in pseudocodice:

Dividendo = 437

Divisore = 7

WHILE (PSUM <= DIVIDENDO) DO

...

END WHILE

SCRIVI contatore

Testing con valori significativi:

* 5/3, 10/3, 2/3, 6/3

#### Challenges

* Consentire all’utente l’introduzione dei due numeri (utilizzando un gadget sviluppato in precedenza)
* Sviluppare la traccia di esecuzione

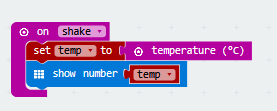
# Introduzione al Physical Computing

## PIEDINATURA MICRO:BIT

<https://makecode.microbit.org/device/pins>

## Sensori interni

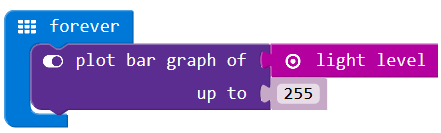
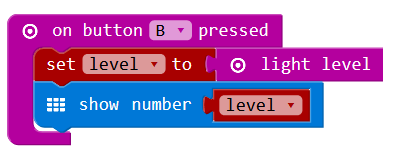
### Termometro



#### Challenge

Visualizzare faccia contenta con t<37, triste fra 37 e 40, distrutta se oltre 40; reset se si scuote

### Light-level sensor



#### Challenge

Visualizzare luna o sole a seconda del livello di intensità luminosa.

## PULSANTE ESTERNO [Pull mode]

Il pull mode di un pin viene automaticamente configurato a seconda della modalità di lettura, read\_analog / read\_digital / is\_touched, rispettivamente come NO\_PULL, PULL\_DOWN, PULL\_UP. Il PULL\_UP sui pin 0-2 è ad alta impedenza (10M), consentendo l’attivazione con touch.

|  |  |
| --- | --- |
| Il seguente utilizza quindi il PULL\_UP con resistore di 10M interno, e può essere attivato touch:  Circuito: R di pull-up interna fra +3 e P0; Switch esterno tra P0 e GND. |  |

Sperimentazione:

* Realizzare e testare un pulsante con della carta stagnola – provare vari collegamenti

## PSYCO-COMPATIBILITY-METER

1. Realizzare pulsante: una persona tocca massa, l’altra il pin di ingresso;
2. Al trigger (stretta di mano), generare numero casuale tra 0 e 100.

## REACTION GAME [Concetto di stato]



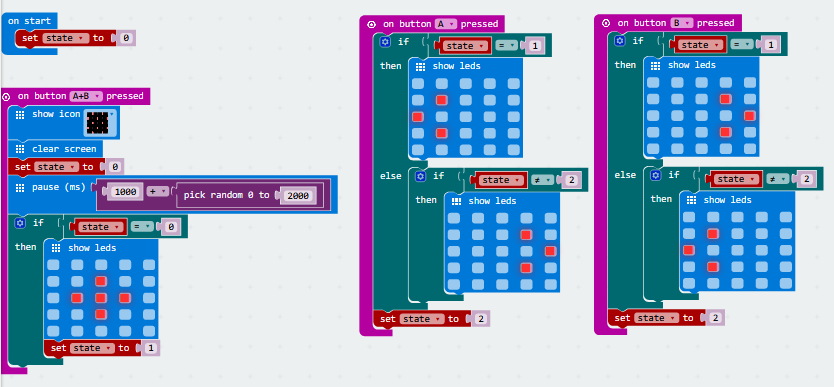
Serale ITIS Paleocapa - 2018

1. Presenta stimolo visivo dopo tempo random 1-3 secondi da evento A+B
2. on A pressed, visualizza freccia verso A; on B pressed, visualizza freccia verso B
3. Interlock fra A e B (se si preme prima A, B deve essere disabilitato, e viceversa) – concetto di stato

Esempio: ReactionGame.hex

#### Challenges

1. Controllare partenza anticipata

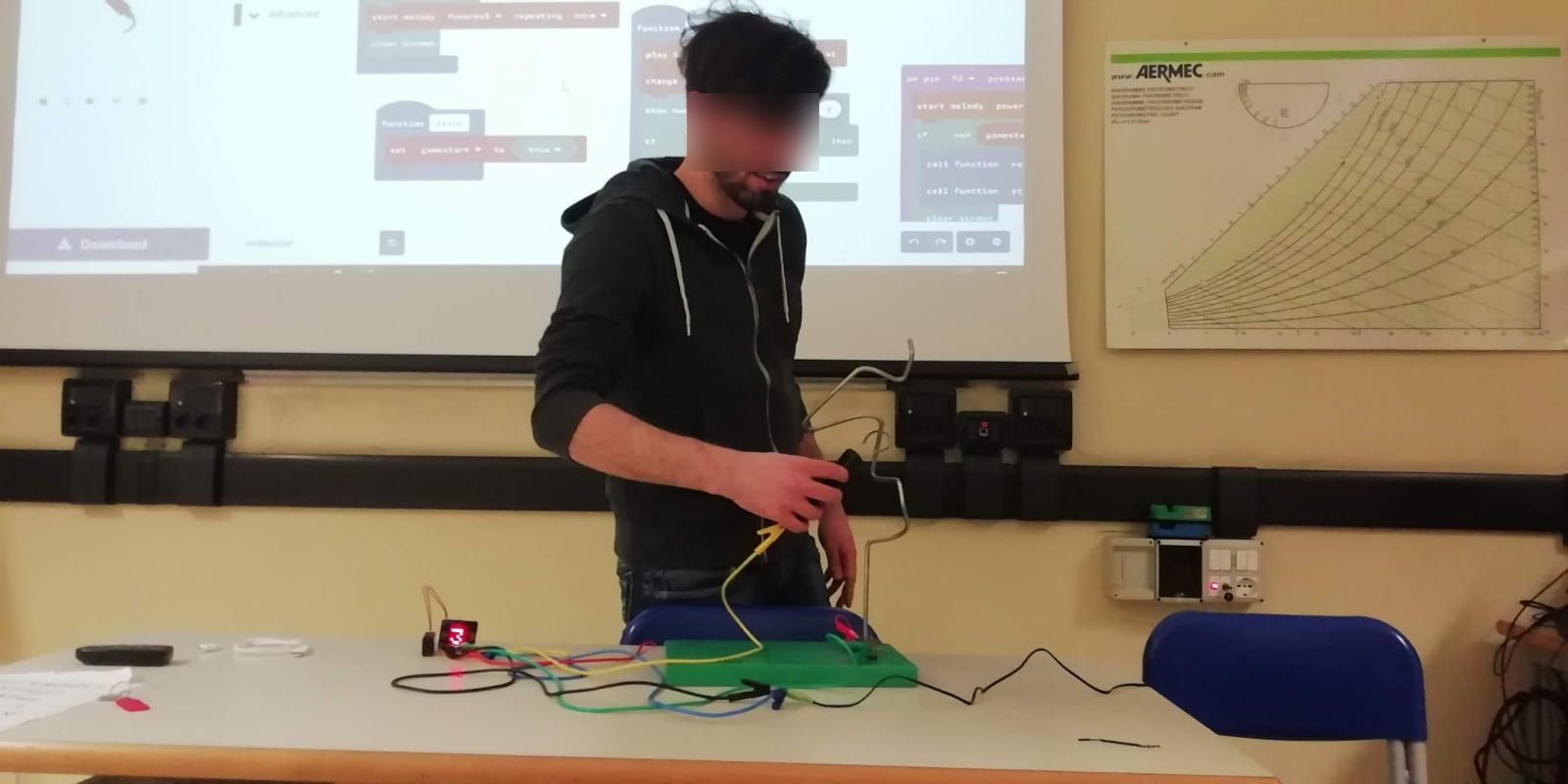


1. Aggiungere punteggio (tenendo conto di tempo e contatti).
2. Randomizzare stimoli visivo e acustico.
3. Realizzare pulsanti fisici e interfacciare (gli ingressi P0-2 hanno resistore interno pull-up di 10M, un valore sufficientemente elevato per consentire l’attivazione tramite tocco (pulsante “touch”): utilizzare l’evento “on Pin P0 pressed”, contatto fra pin0 e massa.
4. Sostituire l’evento di reset A+B con altro evento.
5. Sostituire show leds con plot (per evitare ritardi).
6. STEM: Misura e analisi statistica della velocità di reazione di più giocatori, a uno stimolo visivo versus sonoro; misura della velocità di movimento, differenze fra mano sinistra e mano destra…

Use: 

## WIREBUZZER / MAGIC WAND / FRUSTRATION GAME

Classico percorso in filo conduttore da seguire con un anello senza toccare il conduttore. L’eventuale contatto viene segnalato con un segnale sonoro.



Serale ITIS Paleocapa - 2019

Esempio di “hardware”:



**Rilevazione del contatto (a bassa impedenza):**

|  |  |
| --- | --- |
| I contatti corrispondenti a Button A (sul pin P5) e Button B (sul pin P11) sono disponibili anche sull’edge connector, ed hanno resistori pull-up (interni) di 10k. Tale contatto può quindi essere attivato chiudendo il circuito fra i pin 5 o 11 e la massa, o in modo equivalente premendo i pulsanti A e B. L’attivazione viene quindi poi rilevata, semplicemente, con l’evento “on button A pressed”: |  |

**Gestione suono in uscita:**

|  |  |
| --- | --- |
| con buzzer attivo (pilotato con una tensione continua): |  |
| con buzzer passivo (pilotato con segnale ad onda quadra): |  |

### Challenges:

1. Conteggio numero di contatti
2. Conteggio tempo trascorso fra:
   1. start (indicato da segnale sonoro)
   2. contatto fine gioco

Gestione start con sensore di suono:

* Alimentazione sensore: 3.3V
* Si utilizzano i pins 0-2 (GPIO) come input:
* <https://www.adrirobot.it/sensori/37_in_1/KY-037_KY-038-Microphone-sound-sensor-module.htm>

|  |  |
| --- | --- |
| * Esempio di test del sensore di suono: |  |

1. Start con stimolo visivo (dopo l’esperienza successiva – pilotaggio LED)

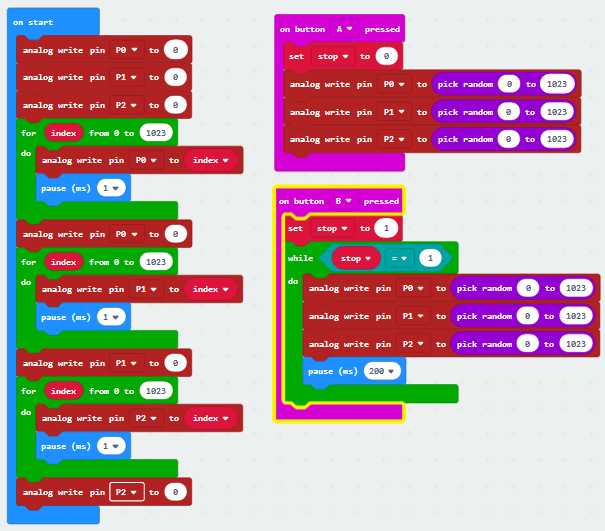
## OUTPUT DI GRANDEZZA ANALOGICA [LED RGB, PWM)

LED RGB: <https://www.lightwaveuk.com/led-technology/how-does-rgb-work/>

|  |  |
| --- | --- |
| LED RGB a catodo comune: |  |

Circuito funzionale: LED RGB a catodo comune, anodi collegati a P0-2 con resistore di 47 ohm.

Lesson strategy: mostrare il comportamento desiderato (file PilotaggioLEDRGB.hex), spiegare l’istruzione “analog write (0-1023)” e chiedere di sviluppare il programma corrispondente.



#### Challenges

* Sviluppare e testare il circuito con tinkercad

### ACCELEROMETRO

|  |  |  |
| --- | --- | --- |
| Come funziona: | | <http://www.explainthatstuff.com/accelerometers.html> <https://www.youtube.com/watch?v=i2U49usFo10> (MEMS) |
| Esempio di rilevazione valori: | |  |
| Utilizzo per gestures: |  | |

## PILOTAGGIO DI UN MOTORE [PWM, accelerometro]

Materiale di riferimento:

* Accelerometro: <http://www.explainthatstuff.com/accelerometers.html> <https://www.youtube.com/watch?v=i2U49usFo10> (MEMS)
* Motore elettrico in DC - <https://www.youtube.com/watch?v=CWulQ1ZSE3c>
* PWM - <https://microbit-micropython.readthedocs.io/en/latest/pin.html>

|  |  |  |
| --- | --- | --- |
| Circuito funzionale: | |  |
| Breadboard (con edge connector): | |  |
|  | |  |
| Software: |  | | |

#### Challenges

* Sviluppare e testare il circuito con tinkercad – visualizzare la forma d’onda su P0
* Sviluppare un prototipo di torcia frontale con regolazione dell’intensità luminosa in funzione della posizione / movimenti della testa

## EGG AND SPOON RACE GAME [accelerometro]

Esempio: [https://makecode.micro:bit.org/examples/egg-and-spoon](https://makecode.microbit.org/examples/egg-and-spoon)

#### Challenges

* Aggiungere punteggio, per esempio quante volte è rimasto al centro su 100 cicli, oppure fino a quando non si preme un pulsante, lo si rovescia…
* Sviluppare un waiter trainer? (equivalente a bicchiere in equilibrio su vassoio)
* Sviluppare un inclinometro (misuratore di pendenza)
* Sviluppare un gioco dove viene visualizzata una strada con curve random, e si deve ruotare il dispositivo per seguirla, con possibilità di variare tipo di strada (curve), velocità….