# Introduction to Computer Programming and Physical Computing Edutainment with micro:bit

This document proposes the development of a series of interactive games with increasing complexity, from pure software to physical computing, aimed at allowing students to discover fundamental coding concepts as independently as possible. More importantly, it seeks to help them acquire computational thinking skills applicable in a broader context, through a constructivist approach throughout. This framework enables students to gain essential computer science skills within an engaging, game-based lab environment, using easy-to-learn programming techniques (such as visual programming) and educational methodologies rooted in experimentation, makers education, and discovery learning.

Pedagogical Guidelines

The teacher should introduce the sequence of projects with a high degree of flexibility, modifying as needed to make the most of students' "discoveries" and suggestions, which cannot be fully anticipated. Through minimal preliminary explanations, targeted hints, and any necessary support, the teacher should maintain each task at a difficulty level that stimulates curiosity and motivation, while avoiding student frustration from being unable to progress. This requires careful monitoring and highly individualized support.

An effective technique is to present the exercises on a learning platform and ask students to immediately share the link to their work in an emulator. This approach enables the teachers to follow students in real-time, minimizing an overly intrusive physical presence, and allows them to quickly identify who needs support, who could benefit from specific guidance, and who may need additional challenges. It also provides opportunities to discuss interesting partial solutions from students with the entire class or to suggest ways to overcome unexpected obstacles.

Encouraging students to prioritize the simplest possible solution ("simple is beautiful") is essential. Friendly contests, where the winning solution is the simplest one that meets the project requirements, can be beneficial, with students themselves determining the winner to foster metacognitive skills that support independent progress.

This document initially provides detailed instructions with complete solution examples, gradually leaving more room for student and teacher initiative. Although developed primarily as a teacher support resource, it can also serve as a student resource when provided progressively, without revealing solutions in advance.

## Reference material

Device Introduction: <http://micro:bit.org/guide/features/>

Development Environment / Simulator: [https://makecode.micro:bit.org/](https://makecode.microbit.org/)

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

Circuit Simulator with micro:bit <https://www.tinkercad.com/things/0O5Lksq72nO-microbit>

## Overview of the Physical and Functional Structure of the micro:bit

Presentation of the Physical Board, specifically:

* Microcontroller: Integrates the CPU, volatile memory, permanent memory, and various I/O (Input/Output) devices.
* Output Devices: LED matrix, loudspeaker.
* Input Devices: Buttons, magnetometer, accelerometer, light sensor, temperature sensor, microphone.

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

Notes on Interacting with the Physical World: The Micro allows interaction with the physical world through additional external sensors and actuators (such as humidity sensors, servomotors, etc.) that can be connected via the external edge connector.

Introduction to the development environment with simulator, including main input and output commands: [https://makecode.micro:bit.org/](https://makecode.microbit.org/)



Heads or Tails [Algorithm, Variable, Selection: IF-THEN-ELSE, Event]

This example should be demonstrated initially, after which students are encouraged to develop a personalized version independently.

**Task**: Develop a program to simulate the flipping of a coin.

Algorithm: If a logical value (true or false) generated "randomly" is true, display "tails"; otherwise, display "heads."

WHEN the device is shaken:

Generate a random value, either true or false.

IF (randomly generated value is true)

THEN show "tails"

ELSE show "heads"

ENDIF

Note: (randomly generated value is true) is equivalent to (generated value).

Prerequisites:

* Show LEDs: Display example images to demonstrate LED control on the Micro
* Demonstrate On Start / On Shake Behavior: Show the basic command show string "Hi" to illustrate how to display text on start-up or when the device is shaken.

|  |  |  |  |
| --- | --- | --- | --- |
| * If Then Else: Use an if statement with a constant (e.g., IF TRUE) to demonstrate how the program behaves. Show the outcome by changing the constant to IF FALSE. | |  | |
| * Use a Logical Variable: | |  | |
| * Pick Random True or False: Use instead of the previous constant (with the event "shake"): | |  | |
| * Alternative codings using a Numerical Variable (more easily applicable in the next exercise): |  | |  |

Challenge: Biased Coin - Ensure that tails appears with a probability greater than 50/100.

Die 1..6: [Multiple IF-THEN-ELSE statements, in sequence and nested]

Task: Simulate the roll of a die.

|  |  |
| --- | --- |
| Basic Version with Show Number: |  |

#### Challenges

|  |  |
| --- | --- |
| * Display the Faces of the Die.   Prerequisite: IF-THEN – show example if necessary: |  |

* Show more efficient solution with ELIF
* Model a biased die
* Model two dice or the sum of two dice
* Check the frequency of occurrence of various events (with two dice: at least one 6, two 6s, two consecutive 6s…)

## Beating heart [repetition – infinite loop]

*Algorithm:*:

FOREVER:

BEGIN

Show big heart

Wait 1 second

Show small heart

Wait ½ second

END

Prerequisite: pause instruction

|  |  |
| --- | --- |
| Code in Block language: |  |

## Beating heart - extension [Event handling – buttons A and B]

Using the following example, briefly explain:

* Concept of Event Handling (ON button A pressed...)
* Concept of Shared Variable Controlled by Events



Tasks:

* Make the heart pulse from the start
* Increase or decrease the heart rate by pressing buttons A or B
* Model cardiac arrest with buttons A and B pressed simultaneously
* Model resuscitation with the shake event

Challenges:

* Ensure that resuscitation can only occur within ten seconds of cardiac arrest; otherwise, death occurs.
* Model arrhythmia (a randomly absent beat, with a probability of 1/5).

|  |  |
| --- | --- |
| Example of basic solution: |  |

## Partial Rock-Paper-Scissors Game [IF-THEN-ELSE in sequence and nested]

Task: Develop an algorithm in pseudocode (pay attention to correct indentation) and program the micro to make a single move of Rock/Paper/Scissors:

* Version with IF in sequence (IF THEN ENDIF IF THEN ENDIF…)
* Version with nested IF (IF THEN ELSE IF… ENDIF ENDIF)
* Compare the efficiency of the two solutions
* Write the pseudocode corresponding to the two cases



## Heads or Tails with coin rolling (difficult extension now, but some students should be able to do it) [Conditional loop]

Task: Display a quick sequence of heads/tails before the final result.

#### Challenges

Slow down the speed of the sequence progressively. Hint: [WHILE (delay < 3000) SHOW…, increase delay, SHOW…, increase delay]

## Scorekeeper gadget [arithmetic expressions, user interface]

Task: Develop a gadget to keep track of the scores of two players (the gadget will be reused, in different forms, in the following exercises):

* Button A increments the score of Player 1, and Button B increments the score of Player 2. Pay attention to the user interface: how to make it clear to the user that they have pressed the button?
* Tilting the micro to the left displays the score of Player A, and tilting it to the right displays the score of Player B.
* Pressing A + B resets the score (starts from 0).

|  |  |
| --- | --- |
| Initializatiom: |  |
| Score Advancement: |  |
| Display Current scores: |  |

#### Challenges

* Manage game win control (for example, the gadget announces when one of the two scores reaches 5).
* Manage matches of three sets. When a player reaches 10, the micro
* declares the set winner and starts a new set. When a player wins three sets, they win the match.
* Display the scores with a row of lit LEDs, where the height indicates the score in the current game. Since there are only five LEDs per column, how to manage 10 points? Propose two different solutions. [Hint: you could use the plot x y instructions...]

Interactive game Rock/Paper/Scissors: user vs. micro with scoring [events]

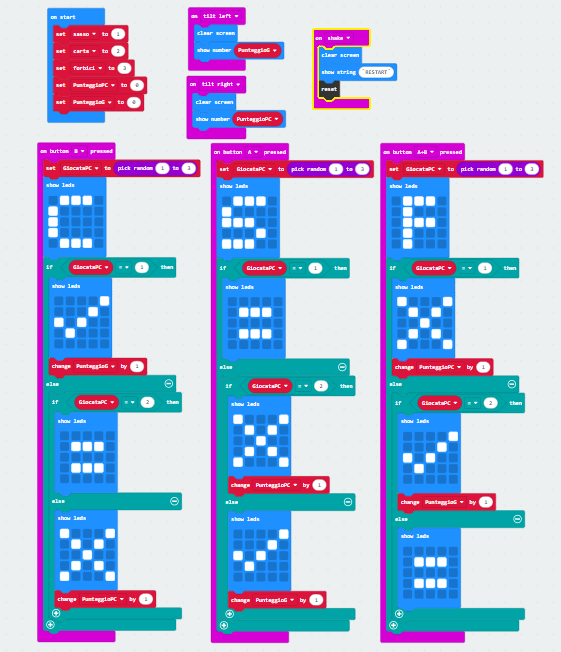
Tasks: Develop the complete Rock/Paper/Scissors game:

* The micro displays the player's move, generates and displays its own random move, shows YOU LOSE/WIN; updates and displays the score.
* The player can access the instructions for the game.
* Tilting left/right shows the current score of the micro/player.

Design the application in pseudocode (pay attention to correct indentation):

* Pay attention to variable names (they should clearly reflect their semantics). It may be useful to have Rock=1, Paper=2..., UserScore, ComputerScore...
* Produce appropriate documentation (analysis, design, coding, testing…).

Example of solution:



#### Challenges

* Check for game over conditions.
* Make the micro cheat by winning more often (20%) than usual.

## Input number gadget 1 [user interface]

Develop a gadget (to be used later) to introduce a number using buttons A and B:

* A +10 (increment tens)
* B +1 (increment units)
* Button A+B ends (confirms) the operation (making it impossible to further modify the entered number).

#### Challenges

* reset on shake
* Allow counting tens/units up to 9 and restart from zero.
* Toggle the direction (up/down) of counting with shake.
* Disable reading with a timeout.
* Increase/decrease with tilting left/right.

## HILO – vanilla version [cycles]

Task: The micro displays a random number between 0 and 100. It then asks the player to guess whether the next generated number will be higher or lower. If the player guesses correctly, they earn a point. The game ends when the player makes an incorrect guess.

#### Produce appropriately commented pseudocode and thoroughly tested code.

#### Challenges

Modify the previous program to increase the player's probability of losing (without exaggerating). This can be achieved by regenerating a new random number whenever the micro secretly detects that the player would win the round.

## Partial HILO – standard version [cycles, integration]

Task: Develop a step of the interactive HILO game (pseudocode and code)

* Generate a random secret number between 1 and 100.
* Read a guess from the user, allowing for +1/+10 and -1/-10 operations with buttons A and B, and A+B to change the sign (integrating the previous gadget).
* Write "guessed," "high," or "low."
* Explicitly outline the testing strategy.

Task: Develop the pseudocode and micro code to complete the HILO game:

* The micro must allow the user a maximum of 5 attempts to guess the number; if the user fails to guess it, the micro declares its victory.
* The buttons must be disabled in case of a win and re-enabled with a shake event to restart the game.

#### Challenges

* Allow the user to enter the maximum number. The computer should then calculate a reasonable number of attempts to grant the user.
* Allow the user to choose the difficulty level.

## Scrolling LEDs [predetermined FOR loops]

Present the instructions to use: plot(x, y), unplot(x, y)

Tasks:

|  |  |
| --- | --- |
| * Make an LED scroll from left to right on the same row |  |
| * Make an LED scroll from left to right on the same row, then on the next row (using a for loop in sequence). |  |
| * Make an LED scroll from left to right on the same row, then iteratively on the next row (using nested for loops). |  |

* Make an LED scroll from top to bottom in the same column, then reverse direction.
* Make an LED scroll diagonally.
* Make an LED scroll downwards (or upwards) at increasing (or decreasing) speeds.
* Make an LED scroll downwards with random speed fluctuations (snowflake effect).

## Timer (and stopwatch)

Task: develop a timer with upcount, downcount, and stop functionalities

Examples:

|  |  |
| --- | --- |
| Upcount from 0 to 4: |  |
| Downcount from 4 to 0: |  |
| Generalize: |  |
| Time: |  |
| Stop when the B button is pressed: |  |

#### Challenges

* Reset when A and B are pressed simultaneously.
* Model a stopwatch with start / stop / pause / continue / lap time / reset.

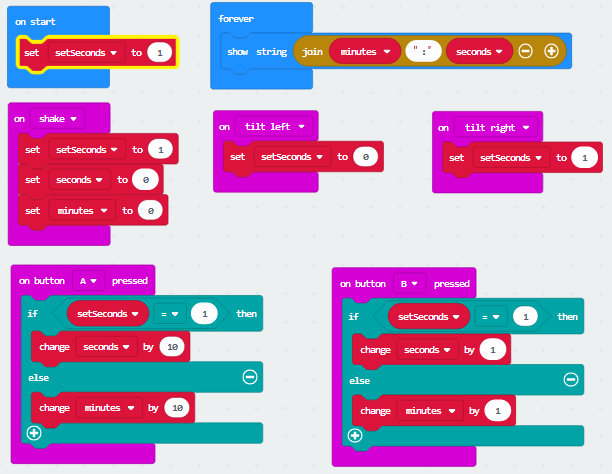
Hints:

* More functions associated with the same button depending on the context:
  + A: start if running = FALSE; stop if running = TRUE
  + B: pause; continue
* Reset: shake
* A + B: lap time

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

Task: develop a gadget that allows setting the minutes and seconds of a timer [file: micro:bit-timerSetting.hex]:

* Display minutes and seconds in the format minutes:seconds
* Initially set to 0:0
* Tilt left to adjust minutes
* Tilt right to adjust seconds
* Press A to increase by 10
* Press B to increase by 1
* Reset by shaking



Warning: the simulator does not work perfectly, as pressing button A also tilts left, and pressing button B also tilts right.

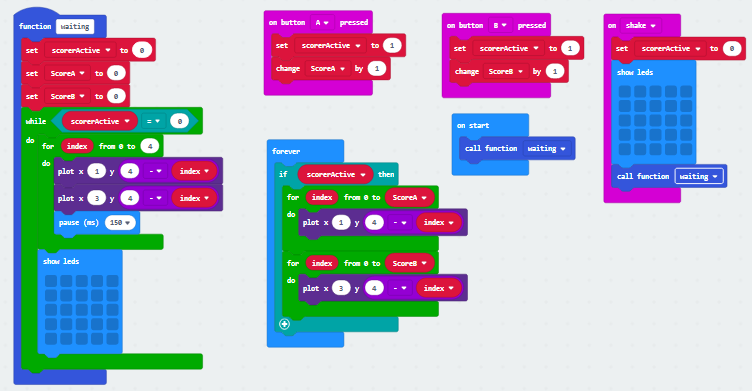
#### Challenges

* Reset the setting to 0 upon reaching 59 (for both seconds and minutes)
* Start the countdown with A+B, signaling when 0:0 is reached
* Use the format mm:ss (e.g., 03:06)
* Design and develop an alternative interface that you consider better

## Visual ScoreKeeper

Develop a visual scorekeeper with the following features:

* Display each player's score with a row of lit LEDs, where each lit LED represents a point.
* Button A increments player 1’s score, button B increments player 2’s score
* Each player’s score is displayed with a column of lit LEDs, with height proportional to the score
* Shaking the device resets the score



File: micro:bit-VisualScoreKeeper.hex

#### Challenges

* Refine the above solution (which starts from 1 each time)
* When the maximum score (5) is reached, a message declares the winner
* Modify the score display for each player:
  + The LED corresponding to the current score remains constantly lit
  + The column of LEDs below should "rise" repeatedly from 0 up to the current score, as if pushing the always-lit LED upward
* ● With each set won, the winning player’s column moves one position toward the center (A moves right, B moves left)
* ● The player who wins the first three sets wins the game, indicated by a message and the flashing of the three columns of the winner

Example: A has won 1 set and has 3 points in the current set. B has won 0 sets and has 4 points in the current set:

A B

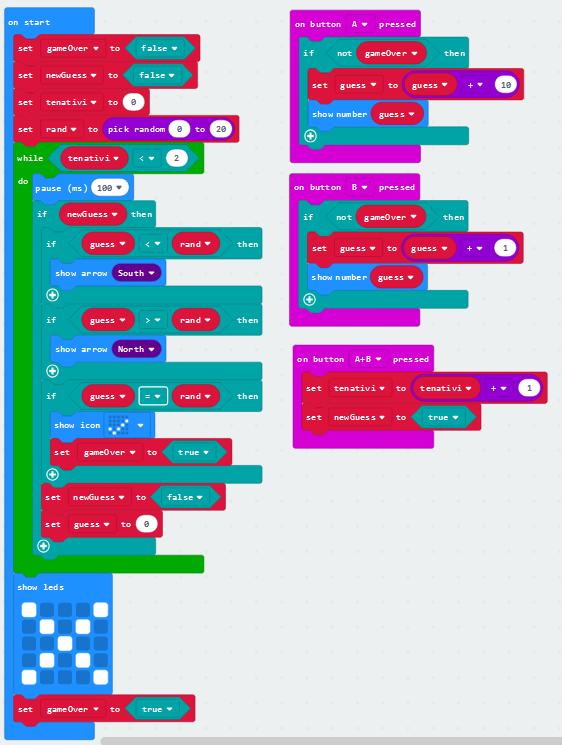
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  | X |
|  | X |  |  | X |
|  | X |  |  | X |
|  | X |  |  | X |

Suggest other improvements to the scorekeeper.

## Complete HILO [while conditional loop]

Use:

* while (attempts < 5) in the onStart event
* Insert pause(100) to allow releasing control to the buttons



[file HILO-while.hex]

* Check what happens without pause(100).
* Improve the code (terminate the loop with AND...) and extend functionality (new game...).
* Write the corresponding pseudocode, appropriately commented.

## Trading simulation [loops, user interface]

Task: A trader performs buying and selling operations in the stock market, earning 50% on positive trades and losing (only) 40% on negative trades. Positive and negative trades are equally likely and occur randomly. Therefore, the trader should, apparently, make a profit... Assuming the trader initially invests €10,000, calculate the remaining amount after 1,000 trades.

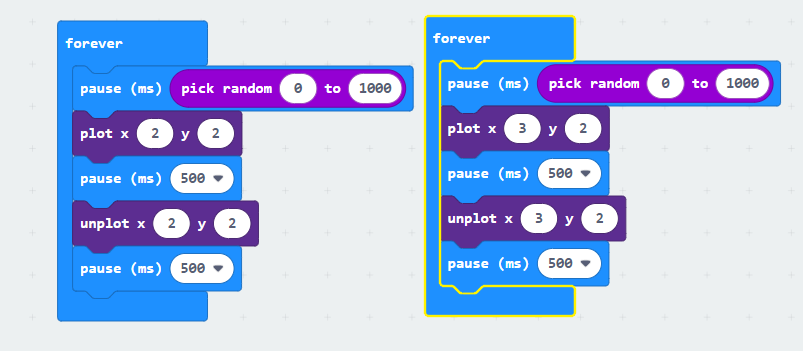
#### Challenge

Visualize in real-time the progress of the total value of their portfolio.

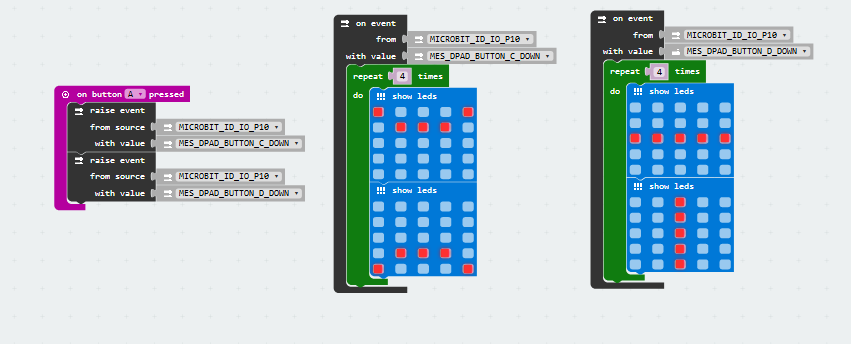
## SlotMachine [concurrency]

Task: Develop a classic-style slot machine with a LED that "scrolls" across three columns at progressively slower speeds until it stops: the player wins when the three LEDs are on the same row.

Managing concurrent "threads":



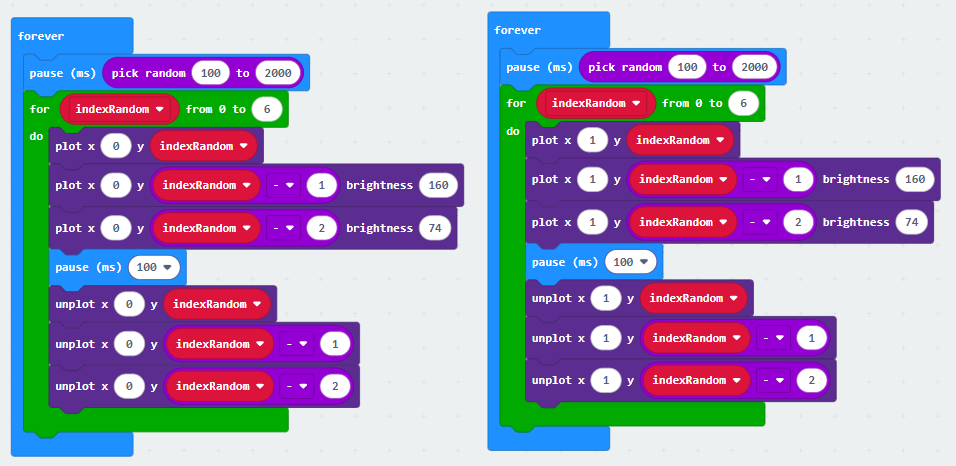
Alternative:

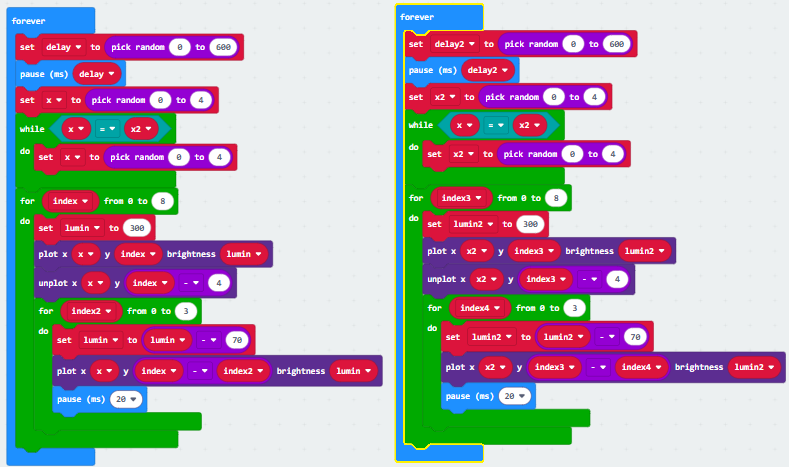


## hooting Stars (Christmas Lights) [Concurrency]

Task: Develop the code to simulate the effect of shooting stars.

Hint: Use multiple forever constructs with random delays (or “run in background” with a while true loop). Examples [file ShootingStars1.hex, ShootingStars1.hex]:





## Generic support exercises (if necessary)

### REPEAT and FOR

* Develop a program that displays "hello" five times using the FOR construct.
* Develop the same program using the REPEAT construct.

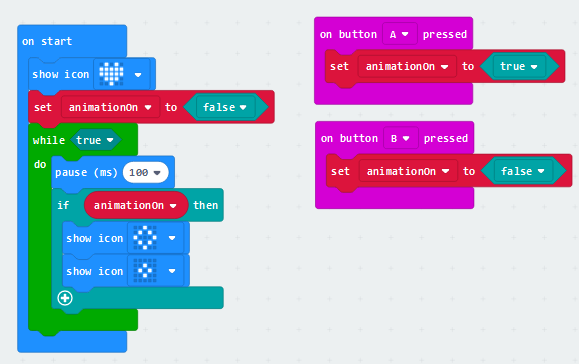
### FOREVER ⬄ WHILE TRUE (Java style) - Two-frames animation

Note: the control variable is set externally to the loop.



### Equivalent implementation more similar to Java

WHILE: body executed "while" the condition is TRUE (note the pause - otherwise it does not release control and gets stuck):



* Experimentally verify what happens without pause(100)
* Brief analysis (functionality, inputs, outputs, variables)
* Correctly indented pseudocode

### Numerical exercises

* Calculate the sum of all integers between 1 and a given (fixed) number.
* Divide et impera (functional decomposition): calculate the sum of all integers between 1 and a number read from the user (with the mechanism seen previously A +10 / B +1 / A+B toggles +/-).
* Calculate the average of the first 15 even numbers.
* Develop a program to write all binary numbers from 1000 to 1111:

1000

1001

1010

1011

1100

...

The program should be easily generalizable, for example, to write all numbers from 1000 to 1555 in base 6 (i.e., 1000 1001 1002 … 1005 1010 1011 1012 ...).

### WHILE: integer division

Calculate the integer division of 437 by 7.

The idea: 23 DIV 7: 7, 14, 21, 28 STOP (!28 <= 23) → 3 with a remainder of 2. The skeleton in pseudocode:

Dividend = 437

Divisor = 7

While (PSUM <= DIVIDENDO) DO

...

END WHILE

WRITE counter

Testing with significant values:

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

#### Challenges

* Have the user input the two numbers (using a previously developed gadget).
* Develop the execution trace.

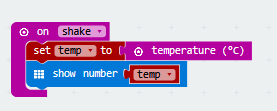
# Introduction to Physical Computing

## MICRO:BIT PINS

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

## Internal sensors

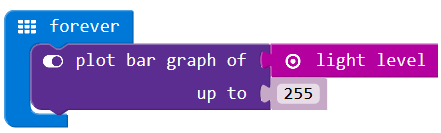
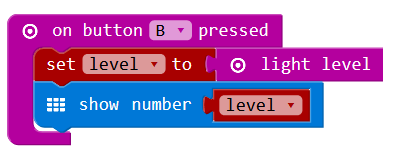
### Thermometer



#### Challenge

Display a happy face if temperature < 37, a sad face between 37 and 40, and a devastated face if over 40; reset if shaken.

### Light-level sensor



#### Challenge

Display a moon or sun depending on the level of light intensity.

## EXTERNAL BUTTON [Pull mode]

The pull mode of a pin is automatically configured based on the reading mode: read\_analog / read\_digital / is\_touched, respectively set as NO\_PULL, PULL\_DOWN, and PULL\_UP. The PULL\_UP on pins 0-2 has high impedance (10M), allowing activation with touch.

|  |  |
| --- | --- |
| The following uses the internal 10M pull-up resistor and can be activated by touch:  Circuit: Internal pull-up resistor (R) between +3V and P0; external switch between P0 and GND. |  |

Experimentation:

* Create and test a button using aluminum foil – try various connections.

## PSYCO-COMPATIBILITY-METER

Create a “button”: one person touches the mass, and the other touches the input pin.

On the trigger (handshake), generate a random number between 0 and 100.

## REACTION GAME [Concetto di stato]



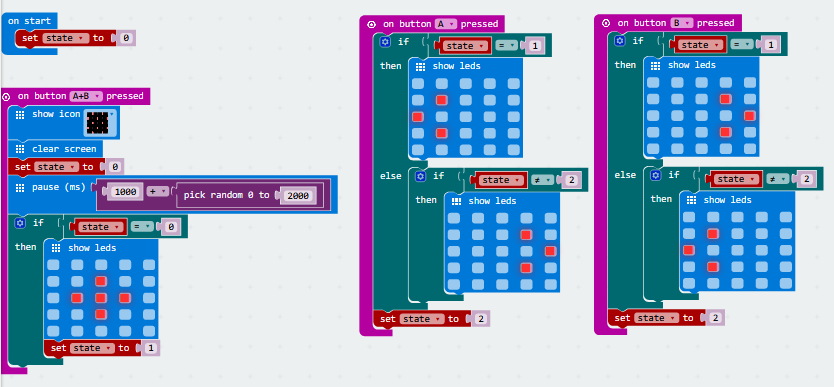
Evening School ITIS Paleocapa - 2018

1. Display a visual cue after a random time between 1-3 seconds following the A+B event.
2. On pressing A, display arrow towards A; on pressing B, display arrow towards B.
3. Implement an interlock between A and B (if A is pressed first, disable B, and vice versa) – introducing the concept of state.

Example: ReactionGame.hex

#### Challenges

Check for early start.

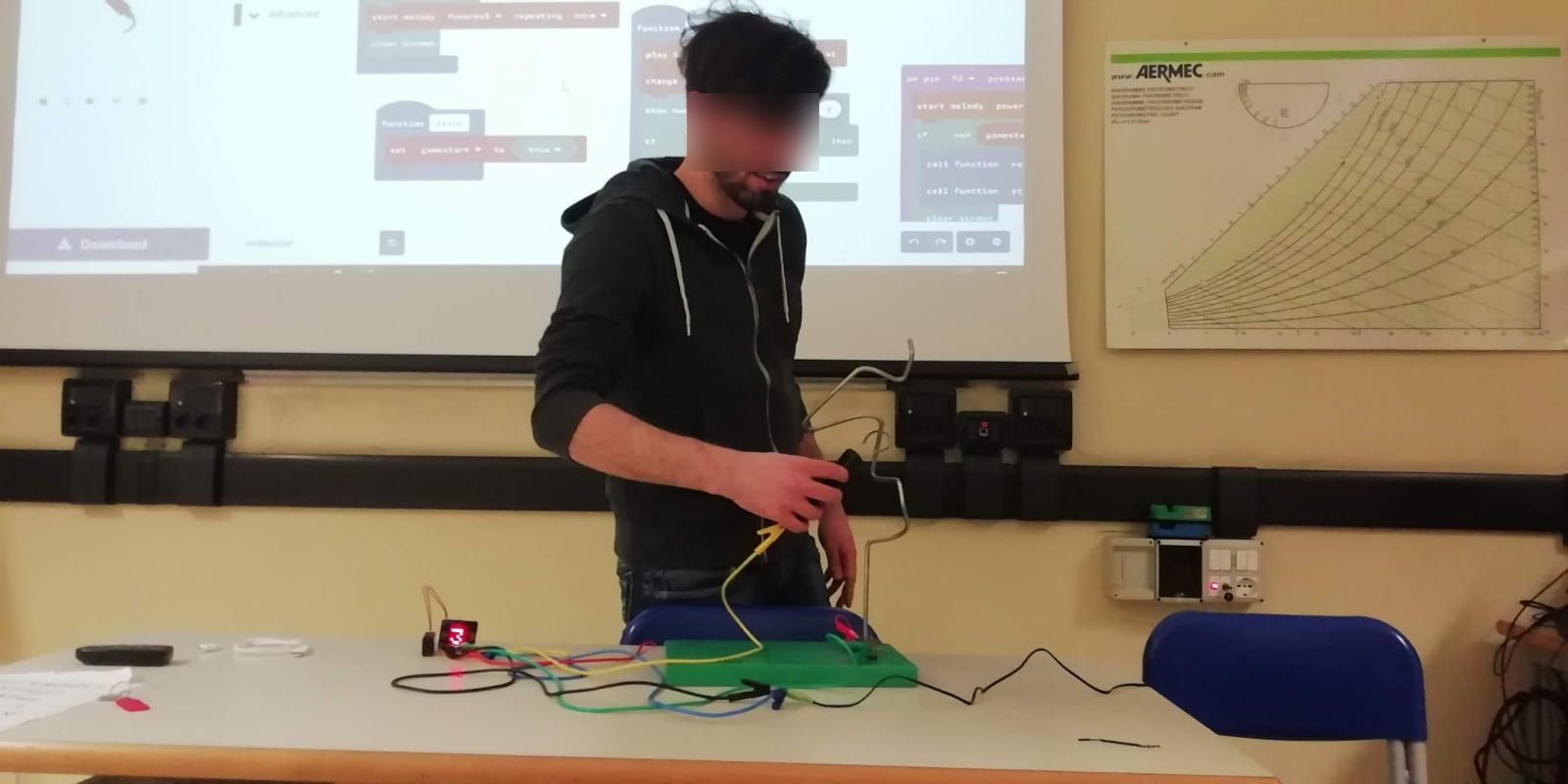


1. Add scoring (considering time and contacts).
2. Randomize visual and auditory stimuli.
3. Create physical buttons and interface them (inputs P0-2 have a 10M internal pull-up resistor, high enough to allow activation by touch—a "touch" button: use the event "on Pin P0 pressed," with contact between pin0 and ground).
4. Replace the reset event A+B with another event.
5. Replace show leds with plot (to avoid delays).
6. STEM: Measure and statistically analyze the reaction speed of multiple players to a visual versus auditory stimulus; measure movement speed, differences between left and right hand...

Use: 

## WIREBUZZER / MAGIC WAND / FRUSTRATION GAME

Classic conductive wire path to follow with a ring without touching the conductor. Any contact is signaled with a sound alert.



Serale ITIS Paleocapa - 2019

Example of “hardware”:



**Contact detection (low-impedance):**

|  |  |
| --- | --- |
| The contacts corresponding to Button A (on pin P5) and Button B (on pin P11) are also available on the edge connector and have 10k internal pull-up resistors. Thus, this contact can be activated by closing the circuit between pins 5 or 11 and ground, or equivalently by pressing buttons A and B. The activation is then simply detected with the event "on button A pressed": |  |

**Sound output management:**

|  |  |
| --- | --- |
| with an active buzzer (driven by a direct current voltage): |  |
| with a passive buzzer (driven by a square wave signal): |  |

### Challenges:

1. Count the number of contacts.
2. Measure the time elapsed between:
   1. Start (with a sound signal)
   2. Contact ending the game.

Sound sensor management:

* Power supply for the sensor: 3.3V.
* GPIO pins 0-2 are used as inputs: [KY-037 / KY-038 Microphone Sound Sensor Module](https://www.adrirobot.it/sensori/37_in_1/KY-037_KY-038-Microphone-sound-sensor-module.htm).

|  |  |
| --- | --- |
| * Example of sound sensor testing: |  |

1. Start with Visual Stimulus (after the subsequent experience – LED control)

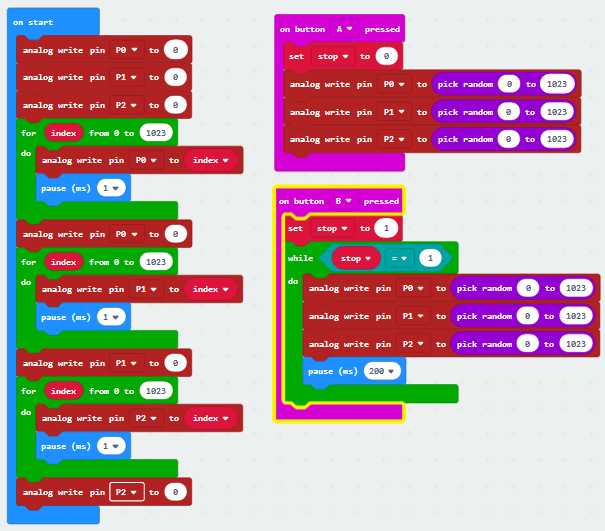
## OUTPUT OF AN ANALOG QUANTITY [LED RGB, PWM)

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

|  |  |
| --- | --- |
| Common cathode RGB LED: |  |

Functional circuit: Common cathode RGB LED, anodes connected to P0-2 with a 47-ohm resistor.

Lesson strategy: demonstrate the required behavior (file PilotaggioLEDRGB.hex), explain the “analog write (0-1023)” instruction, and ask students to develop the corresponding program.



#### Challenges

* Develop and test the circuit with Tinkercad.

### ACCELEROMETRO

|  |  |  |
| --- | --- | --- |
| How it works: | | <http://www.explainthatstuff.com/accelerometers.html> <https://www.youtube.com/watch?v=i2U49usFo10> (MEMS) |
| Example of values detection: | |  |
| Use for gestures: |  | |

## CONTROL OF A MOTOR [PWM, accelerometer]

Reference material:

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

|  |  |  |
| --- | --- | --- |
| Functional circuit: | |  |
| Breadboard (with edge connector): | |  |
|  | |  |
| Software: |  | | |

#### Challenges

* Develop and test the circuit with Tinkercad – visualize the waveform on P0
* Develop a prototype of a headlamp with adjustable brightness based on the position/movement of the head.

## EGG AND SPOON RACE GAME [accelerometer]

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

#### Challenges

* Add scoring, for example, how many times it has remained centered over 100 cycles, or until a button is pressed, it gets flipped...
* Develop a waiter trainer (equivalent to a glass balanced on a tray).
* Develop an inclinometer (tilt meter).
* Develop a game where a road with random curves is displayed, and the device must be tilted to follow it, with the option to vary the type of road (curves), speed, etc.