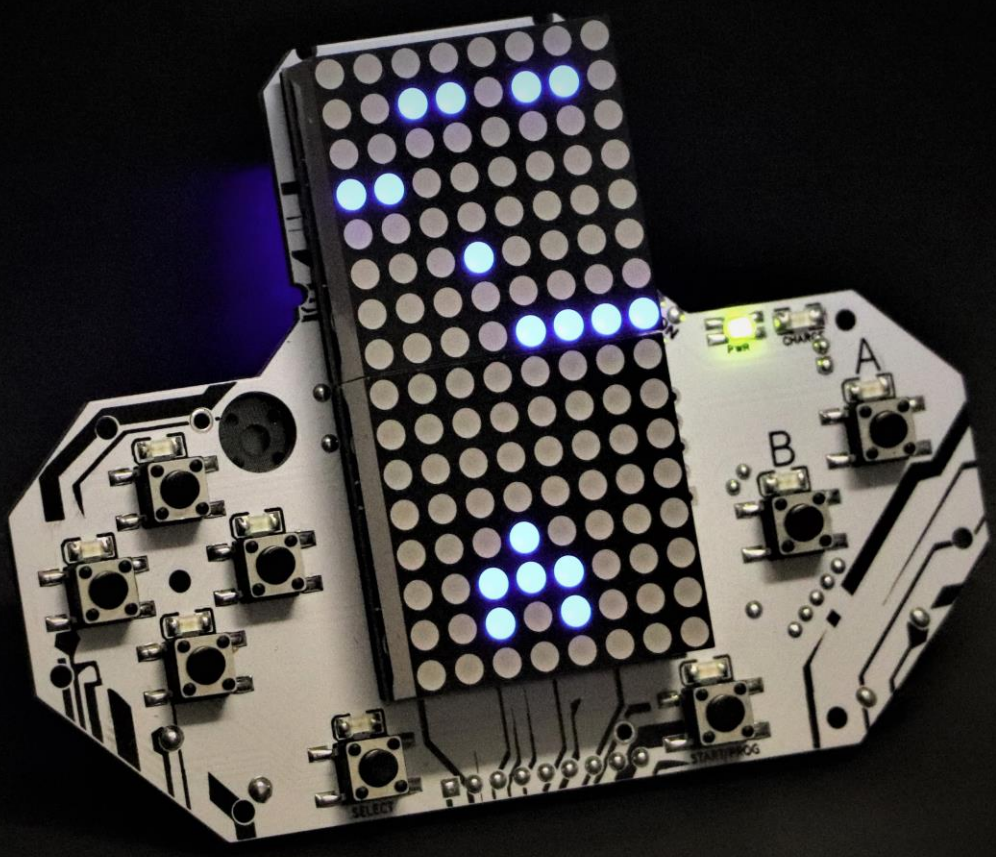
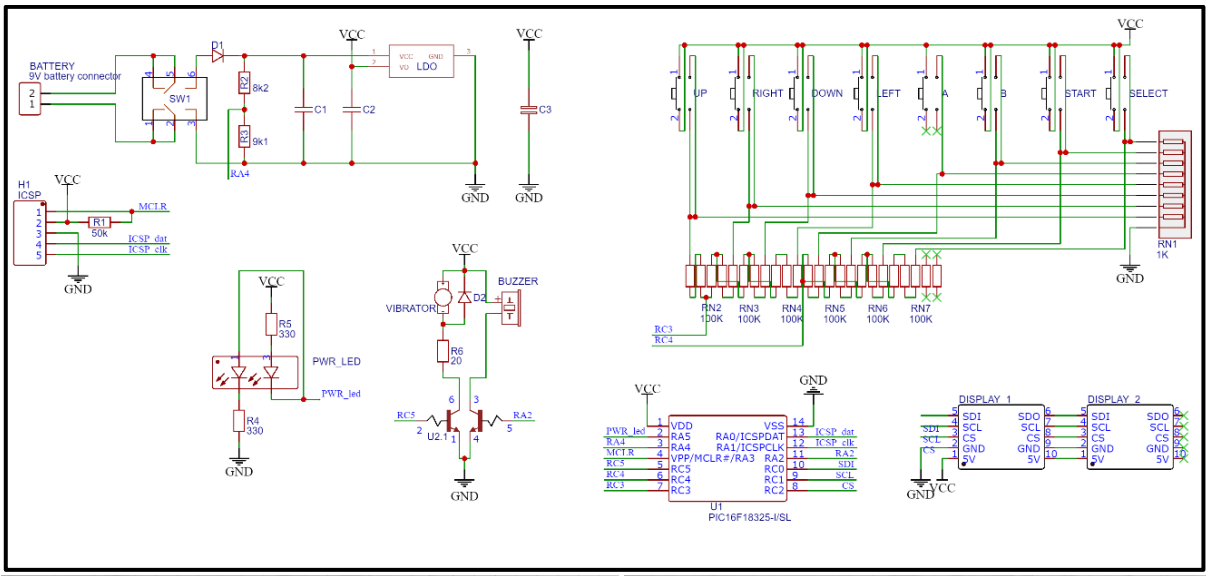
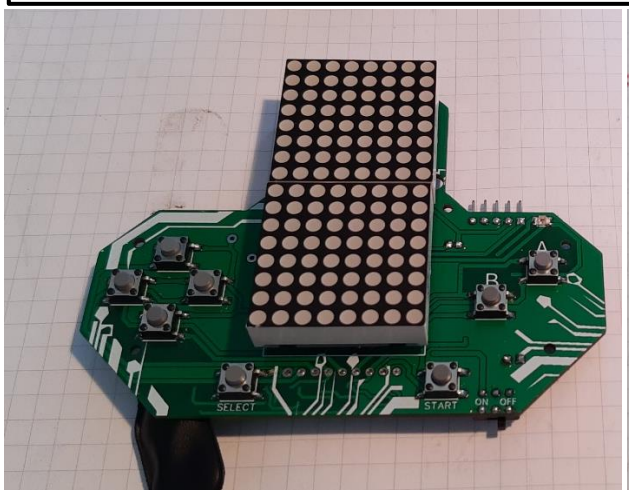
GameCode

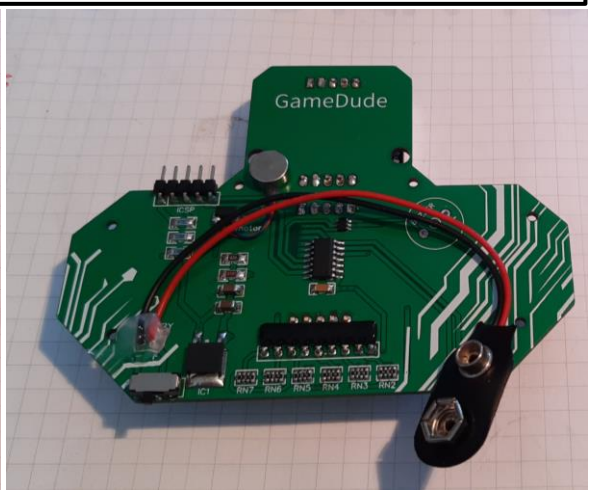


Hi, I'm Jef De Bruyne! Teaching embedded programming to 14 to 18 year olds isn't always easy. Finding fun projects to keep the students' motivation high for this abstract subject is key. You have to find a balance between cool projects that spark excitement but aren't too complicated hardware-wise so the students understand what the software is doing exactly. At the same time, these projects need to cover enough of the school curriculum. Arduino is well-known, but in a classroom environment, a breadboard is not an option for long term projects and PCB design/manufacturing not always possible. I wanted to create a ready-to-go project for the students without the need of wiring. A project that's challenging yet accessible, covers a large part of the curriculum and is scalable in difficulty. That's how I came up with the GameCode.

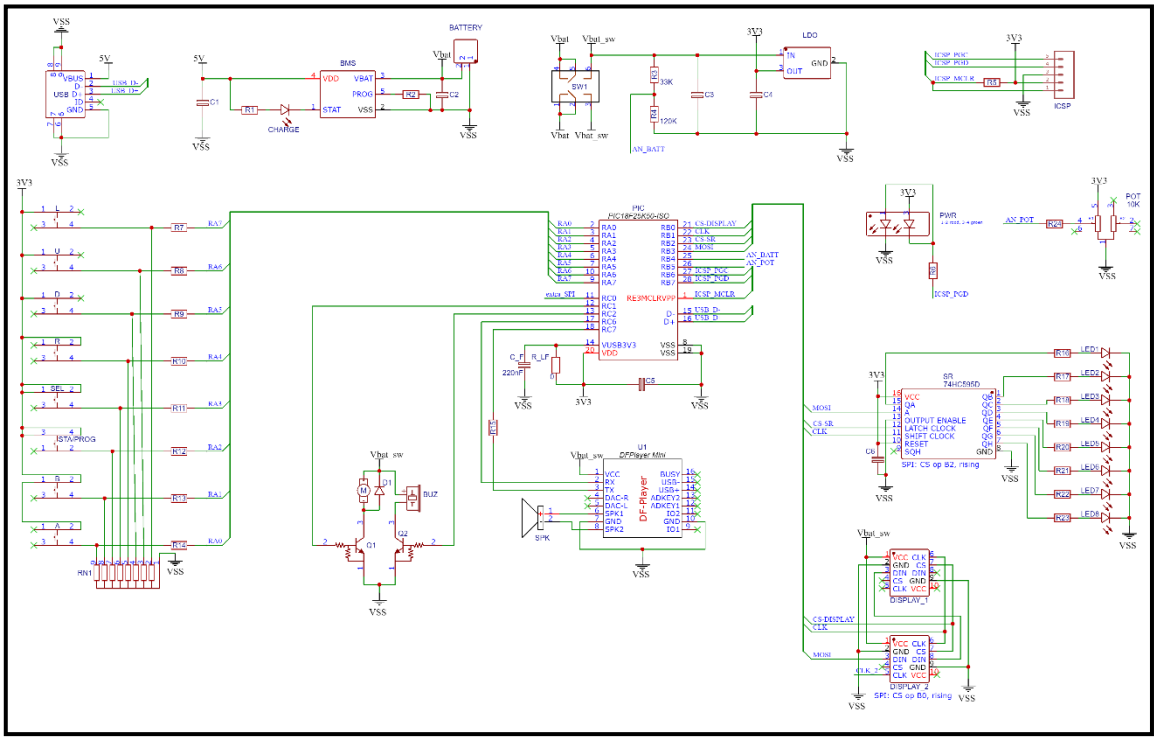
The idea was to create a little handheld game console, which students can use to invent and program their own games. I started researching, and designed the first prototype. I let my students decide the name of the project. And thus GameDude was born.



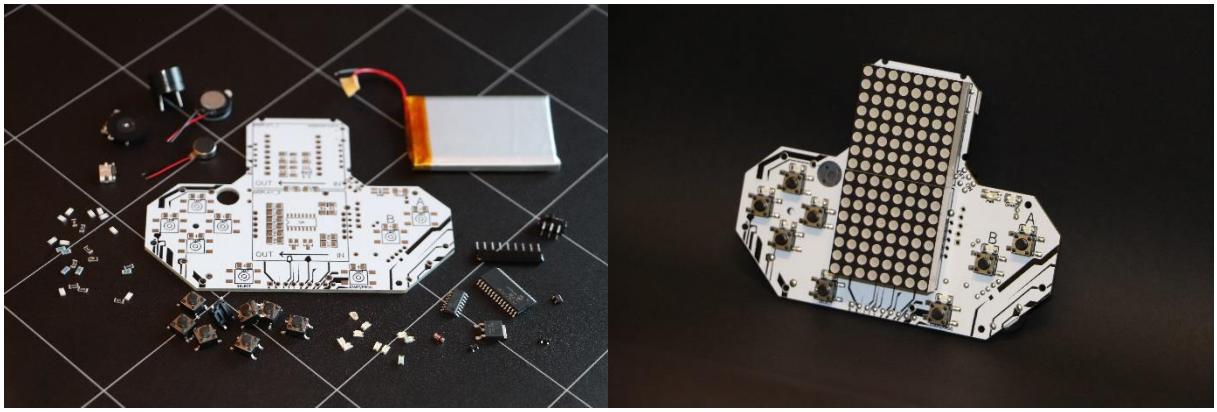




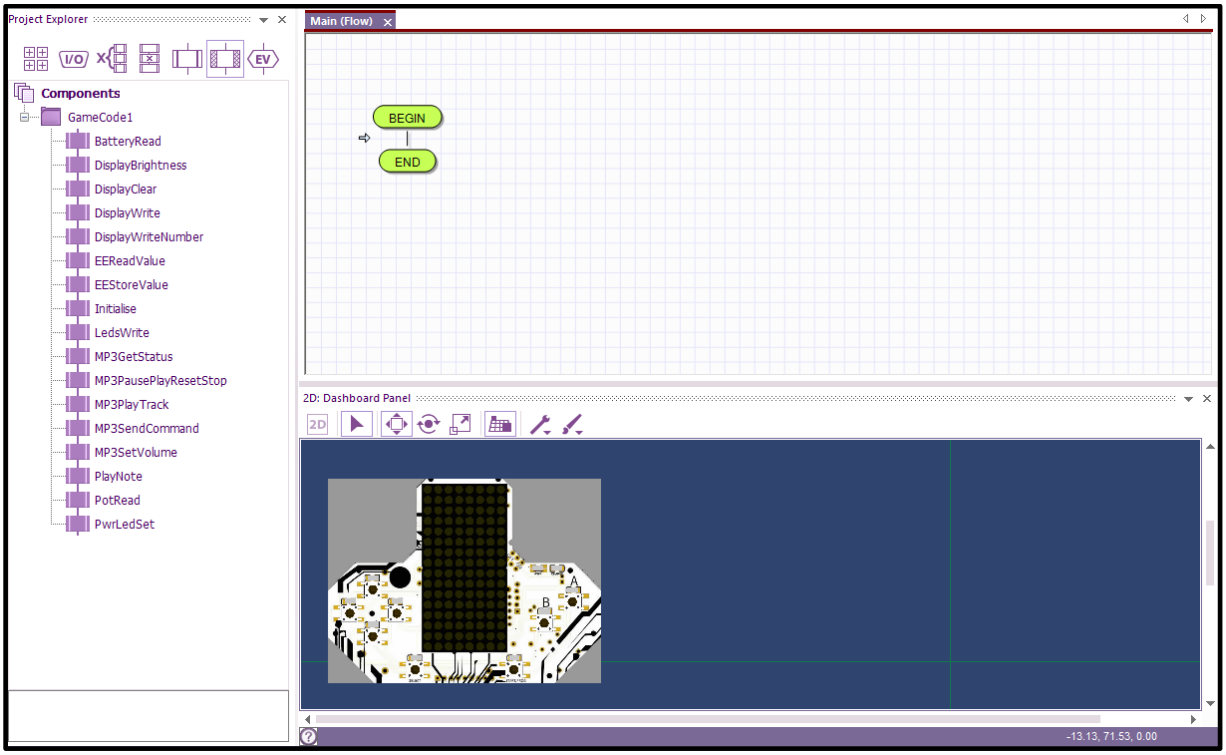
This first version was very simple; two led-dot matrices, bi-colour led, eight buttons, buzzer and a vibration motor. Software had to be uploaded using a programmer. I started further development to unlock the full potential of this product.



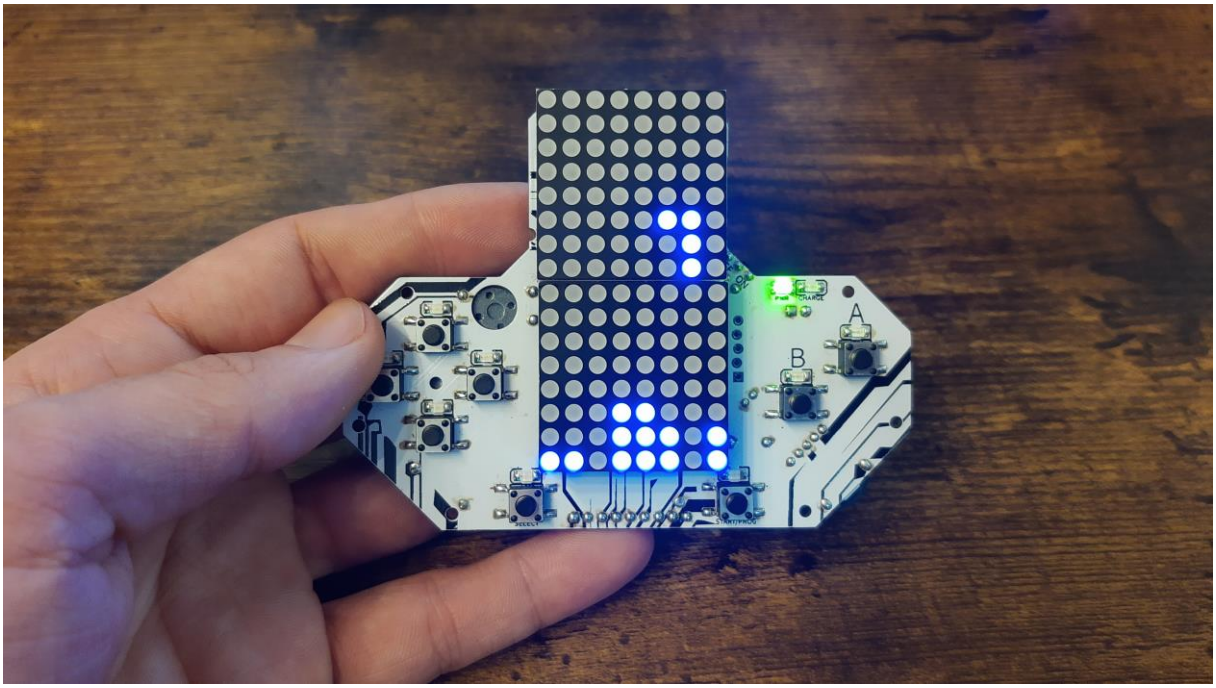
Three prototypes later, a lot had changed. A rechargeable battery was added, together with a usb port (and bootloader) to charge the device and upload software. I included a mp3-module, leds for each of the buttons and much more. Now, the hardware was on point and the name changed to GameCode.



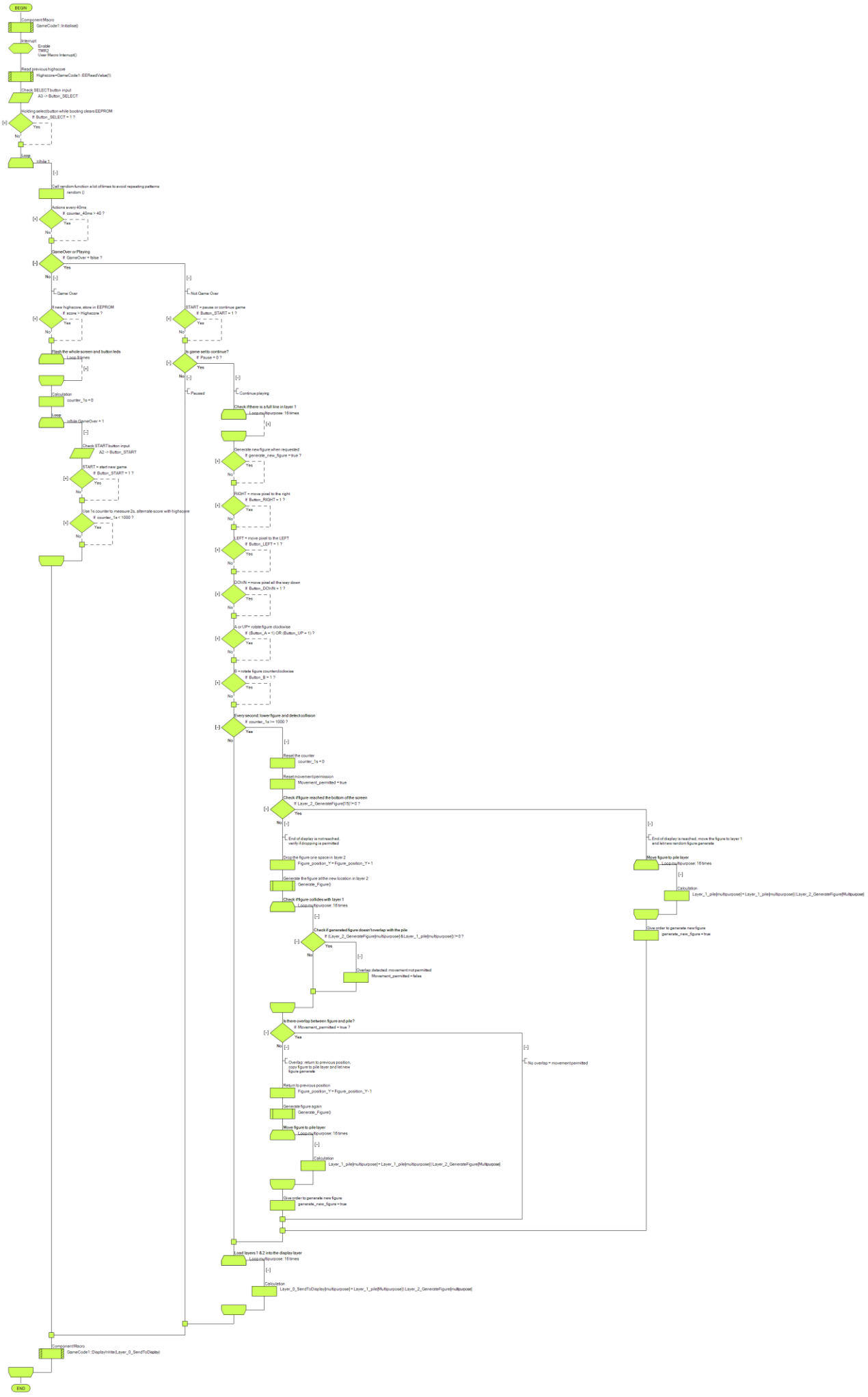
Next up was ease of programming. I was already teaching my students to program using Flowcode, so I reached out to Matrix and they were happy to help! Together with the Flowcode developers (special thanks to Ben Rowland) we created special component macro's that were implemented in the Flowcode software. The macro's greatly simplified some of the more complex code that would be needed for certain features of the device. Using these, even the students with little programming knowledge will be able to program simple games for GameCode!



Using Flowcode I created a Tetris game to demonstrate the possibilities. Theme music can be played, display brightness altered, vibration feedback when completing a row, high scores saved and so on.



See the next page for an overview of the code. I hope you find GameCode as interesting and fun as I had developing it! If you want to learn more about it, pay a visit to www.pagkat.be.


BEGIN
ComponentMacro
GameCode1::Initialise()
Interrupt
Enable
TIMR2
User Macro Interrupt()
Read previous highscore
Highscore=GameCode1::EEReadValue(1)
Check SELECT button input
A3 -> Button_SELECT
Holding select button while booting clears EEPROM
If Button_SELECT = 1 ?
Yes
No
Loop
while 1
Call random function a lot of times to avoid repeating patterns
random ()
Actions every 40ms
If counter_40ms > 40 ?
Yes
No
GameOver or Playing
If GameOver = false ?
Yes
No
Game Over
If new highscore, store in EEPROM
If score > Highscore ?
Yes
No
Flash the whole screen and button leds
Loop 5 times
Calculation
counter_1s = 0
Loop
while GameOver = 1
Check START button input
A2 -> Button_START
START = start new game
If Button_START = 1 ?
Yes
No
Use 1s counter to measure 2s, alternate score with highscore
If counter_1s < 1000 ?
Yes
No
Not Game Over
START = pause or continue game
If Button_START = 1 ?
Yes
No
Is game setto continue?
If Pause = 0 ?
Yes
No
Paused
Continue playing
Check if there is a full line in layer 1
Loop multipurpose: 16 times
Generate new figure when requested
If generate_new_figure = true ?
Yes
No
RIGHT = move pixel to the right
If Button_RIGHT = 1 ?
Yes
No
LEFT = move pixel to the LEFT
If Button_LEFT = 1 ?
Yes
No
DOWN = move pixel all the way down
If Button_DOWN = 1 ?
Yes
No
A or UP= rotate figure clockwise
If (Button_A = 1) OR (Button_UP = 1) ?
Yes
No
B = rotate figure counterclockwise
If Button_B = 1 ?
Yes
No
Every second: lower figure and detect collision
If counter_1s >= 1000 ?
Yes
No
Reset the counter
counter_1s = 0
Reset movement permission
Movement_permitted = true
Check if figure reached the bottom of the screen
If Layer_2_GenerateFigure[15] != 0 ?
Yes
No
End of display is not reached,
verify if dropping is permitted
Drop the figure one space in layer 2
Figure_position_Y = Figure_position_Y + 1
Generate the figure at the new location in layer 2
Generate_Figure()
Check if figure collides with layer 1
Loop multipurpose: 16 times
Check if generated figure doesn't overlap with the pile
If ((Layer_2_GenerateFigure[multipurpose] & Layer_1_pile[multipurpose]) != 0 ?
Yes
No
Overlap detected: movement not permitted
Movement_permitted = false
Is there overlap between figure and pile?
If Movement_permitted = true ?
Yes
No
Overlap: return to previous position,
copy figure to pile layer and let new
figure generate
Return to previous position
Figure_position_Y = Figure_position_Y - 1
Generate figure again
Generate_Figure()
Move figure to pile layer
Loop multipurpose: 16 times
Calculation
Layer_1_pile[multipurpose] = Layer_1_pile[multipurpose] | Layer_2_GenerateFigure[Multipurpose]
Give order to generate new figure
generate_new_figure = true
No overlap = movement permitted
End of display is reached, move the figure to layer 1
and let new random figure generate
Move figure to pile layer
Loop multipurpose: 16 times
Calculation
Layer_1_pile[multipurpose] = Layer_1_pile[multipurpose] | Layer_2_GenerateFigure[Multipurpose]
Give order to generate new figure
generate_new_figure = true
Load layers 1 & 2 into the display layer
Loop multipurpose: 16 times
Calculation
Layer_0_SendToDisplay[multipurpose] = Layer_1_pile[Multipurpose] | Layer_2_GenerateFigure[multipurpose]
ComponentMacro
GameCode1::DisplayInit(Layer_0_SendToDisplay)
END