DESIGN OF A CONNECTED TOY FOR BABIES

Author: Sanz Giner, Alfonso
Supervisor: Álvaro Pérez Bello
Collaborating Entity: ICAI – Universidad Pontificia Comillas & Altair Engineering Inc.

ABSTRACT

This project comprises the development of a prototype of a new **toy** not only for entertainment purposes, but also for the cognitive development of infants. All of that while proving possible the design of a connected device capable of tracking all the events of the players for future study and allowing the progressive increase in difficulty in proportion to the player results.

The connected capabilities allow the monitoring of the device inputs, outputs and state (received via MQTT) in Altair's SmartWorks **IoT platform**; and – thanks to the platform programming capabilities – the computing of the games' "state machine" can be performed cloud-based, with HTTP requests to the platform API to access the variable values and compute the next instructions to send to the device. This allows the easy implementation of new games by adding game modes programmed in Python instead of modifying the integrated programming of the device.

INDEX TERMS

Game, programming, circuit, code, Internet of Things (IoT), MQTT, HTTP, components, electronics, Arduino, project, construction, prototype.

I. INTRODUCTION

Nowadays there are countless electronic toys for babies, but all of them count with the same limitations in terms of **playability**, variety, and duration of gameplay. These toys intend to be flashy, with sounds and lights, but very simple so that it is easy to get bored of them and move on to the next one. To solve this dilemma, the prototype designed in this project should allow limitless **scalability** in terms of new games and greater difficulty thanks to its IoT capabilities.

The device is a 12x12x12 cm cube-shaped wooden box with a transversal cut and empty interior for the **electronic components**, which include the microcontroller, PCB, port expander, various sensors – such as an accelerometer/gyroscope and buttons – and outputs to guide the player through the game, which are 6 coloured LEDs (one in each side) and a speaker.

Multiple **technologies** – including the programs, electronics and communication protocols that are mentioned in this document – are used to ensure the simultaneous functionality of the playability, monitorization and IoT communication.

The development of the project is divided into 5 main **tasks**:

- Planification, investigation and preparation.
- Physical construction of the prototype.
- Programming of components.
- IoT development.
- Documentation.

The **methodology** implemented will require simultaneous progress in all tasks at the same time, as discovering/testing components and utilities will require continuous adaptation of the approach.

II. DESCRIPTION OF TECHNOLOGIES

As for the **electronics components** included in the final design, the list is the following:

- Microcontroler (NodeMCU ESP8266)
- LED buttons (Arcade Illuminated Buttons)
- I2C Port Expander (MCP23017)
- Accelerometer/gyroscope (MPU6050)
- Speaker/buzzer (Mini Metal Speaker 8Ω)
- Wooden box (Artemio VIBB19)
- Printed Circuit Board (PCB), pins and wiring

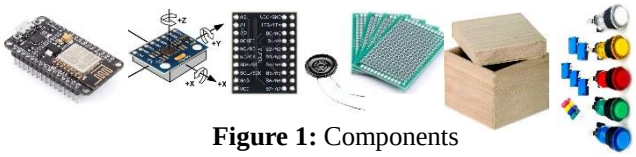

Figure 1: Components

The microcontroller selected is similar to an Arduino Nano, but with a Wi-Fi module that makes it very common in IoT projects. It can be programmed in various frameworks (NonOS, RTOS…), but Arduino's SDK is the one used due to previous experience and its popularity, with therefore superior access to documentation.

The 6 LED buttons are positioned on each face of the cube and are of a different colour (blue, yellow, green, red and 2 white for the top and bottom face). Since each LED and button require 2 connections – inputs and outputs – to the microcontroller a port expander is used to increase the amount of IO pins of the microcontroller.

The port expander works through I2C – like the accelerometer – which allows for the use of only 2 pins for controlling and communicating with the components (SCL and SDA). And it includes PULLUP resistors that remove the need for additional resistors for the buttons' voltage.

The accelerometer can be used as an inclinometer by using the values of the acceleration in each direction (X, Y and Z) and calculating the angle with basic trigonometry, or simply detecting the direction with higher acceleration as the downward face, as it is detecting the gravity. This works providing it is not being moved with an acceleration higher than the value of gravity. More complex calculations can be used by calibrating the device when facing the correct direction and using the values of the gyroscope to detect the new position based on the velocity and time elapsed (public libraries are already capable of this), but it is not necessary for the prototype as it works perfectly for its purpose as it is.

Finally, the speaker is capable of a wide range of frequencies (~600-10KHz) which implies it can be used to reproduce basic melodies for victory/timeout.

Other tools or minor components have been used for the construction: Tin and soldering iron for the PCB, a protoboard for testing, driller and drill bits for the wooden box, gun and hot glue, cable cover and insulating tape.

The **programs/software** used for the project are:

- Visual Studio Code (instead of Arduino IDE due to its compatibility and features)
- Platform IO (VSCode tool for programming of integrated systems)
- GitHub (version control)
- Altair SmartWorks IoT (IoT platform)
- EasyEDA (PCD design)

III. CONSTRUCTION OF THE PROTOTYPE

A. PHYSICAL CONSTRUCTION

The physical construction of the project requires 3 subtasks:

- Drilling the wooden box
- Soldering of the PCB
- Wiring of all the components

The wooden box must be drilled on each side with 28 mm holes to insert the LED buttons with an additional 3 mm hole in the opening for the power/debugging cord.

The soldering is the most time-consuming task as header pins will be used to allow for the easy connection and disconnection of components. The design of the PCB was made with EasyEDA as shown in the next illustration:

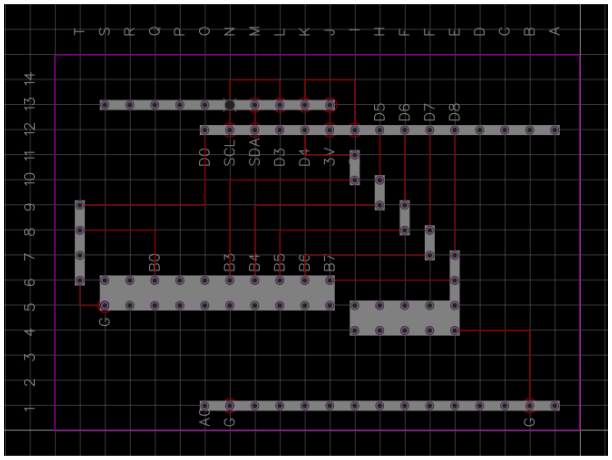

Figure 2: PCB design

The PCB is soldered on both sides, one includes the connection for the microcontroller and the other the port expander and all the LEDs/buttons:

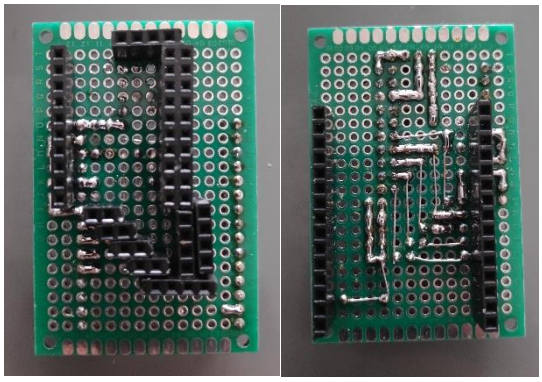

Figure 3: PCB soldered on both sides

Most components also require some soldering for the cable and pins that are going to be connected to the header pins of the PCB. But this document – as a summary – will not detail these aspects; the same goes for the wiring, which is duly explained in the main document. However, the next table shows the different pins of the microcontroller (blue) or port expander (yellow) to which are connected the LED, buttons, and accelerometer:

The following image shows the result of connecting all buttons, PCB, speaker and accelerometer inside the box (the last one is glued to the upper face):

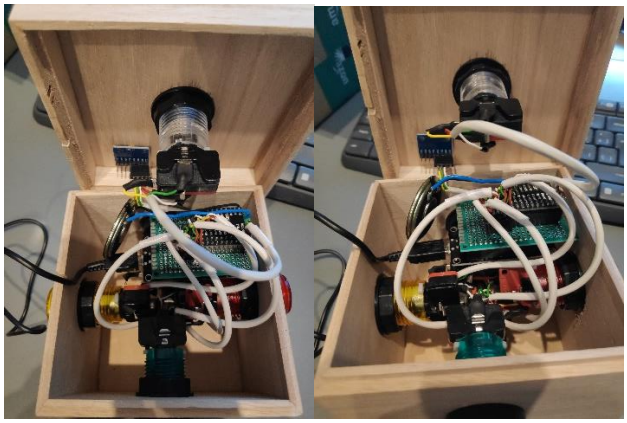

Figure 4: Full wiring

As a first prototype, the device should prove the viability of the idea and it is not a commercial MVP design, as the box is significantly larger than appropriate for small infants' hands and includes a cable for power and programming/debugging instead of a battery. Nevertheless, the construction is solid and reliable, as the components can resist a fall without disconnecting or breaking.

B. PROGRAMMING

The programming of the NodeMCU requires testing the components' functionality, developing a library/class for its simple control and finally designing the main program with the state machine, the full communication with the IoT platform and the basic game mode to test.

Once again, the full programming explanation is available in the main document, but the next section briefly explains the game modes and libraries used and programmed.

Wiring	Connector	White	Blue	Green	Yellow	Red	White	MPU6050
red	COM (down)	GND	GND	GND	GND	GND	GND	VCC
white	LED (red)	GND	GND	GND	GND	GND	GND	GND
yellow	NO (front)	MCP8	MCP11	MCP12	MCP13	MCP14	MCP15	SDA (D1)
green	LED (rear)	D0	D4	D5	D6	D7	D8	SCL (D2)
		regular	reverse	reverse	regular	regular	reverse	

Figure 5: Table of pin connections

As previously mentioned, the programming is done in VSCode with PlatformIO using Arduino SDK, meaning the language employed is C++. The next list briefly explains the libraries used and programmed for the project:

Integrated libraries:

- Arduino.h (Arduino functions for PlatformIO)
- Serial.h (well known for communication with the PC via UART protocol)
- Wire.h (I2C communication for the control of the MCP23017 and the MPU6050)

Public libraries:

- Adafruit_MCP23X17.h (control of the port expander)
- ESP8266WiFi.h (definition of the Wifi client)
- PubSubClient.h (by knolleary, definition of the MQTT client functions)
- ArduinoJson.h (by bblanchon, for parsing the messages received and sent in JSON format)

Private libraries (programmed):

- pitches.h (list as constants the frequencies for the speaker)
- spkControl.h (class for reproducing melodies, controls the timing and list of melodies)
- Adafruit MPU6050 (readings of the accelerometer and integrated logic)
- credentials.h (holds the credentials and topics necessary for the MQTT connection)
- **wifi-mqtt.h** (contains the whole communication with the platform and allows to update the properties of the IoT database via MQTT and receive instructions as messages)
- PINS.h (lists of pins connections)
- **game.h** (main programming of the state machine)

The main program is very simple and thanks to the object-oriented programing only requires initializing the wifi-mqtt and game classes, and – in the main loop – it checks for new messages/instructions and depending on the game mode selected executes the corresponding function.

There are 3 main **game modes**:

The **testing** mode allows checking the correct behaviour of all the devices and the connection with the IoT platform. Essentially, every time a button is pressed it turns on and a message is sent, updating the corresponding property in Altair SmartWorks, the same happens when changing its orientation.

The **main** mode is a simple game with progressive difficulty capabilities. A number of buttons (from 1 to 5) – depending on the difficulty selected in the IoT platform – turn on randomly and the player must find them and press them at the same time. There is a time limit that can also be set on the platform. As before, all the events are detected and updated via MQTT messages.

The **slave** mode is the most important, as it allows the device to follow the instructions sent by the platform. This means the computing and state machine is completely done cloud-based as the device will only be in a state of waiting for instructions or executing them. This allows for all the programming of games and scalable implementation of new games to be done via the IoT platform without having to modify further the internal code executed in the device.

These modes serve, not only as a proof of concept of the main objectives of this project (which are to make a toy with monitoring and progressive difficulty), but also introduce – with the slave mode – the possibility of developing an entertaining and education platform full of easily programable games and capable of cognitive development studies based on the events tracked and the progress of the player. These possibilities will be defined in more detail in the last section (IV. ANALISIS, CONCLUSIONS AND FUTURE DEVELOPMENT).

C. IoT DEVELOPMENT

The device is connected with Altair SmartWorks, Altair's IoT platform, and its main utility is AnythingDB, the platform's database, which is organized with the following structure:

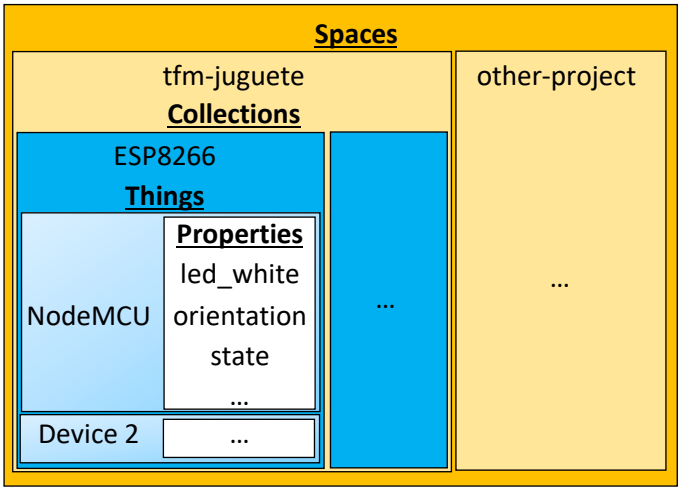

Figure 6: AnythingDB's structure

Each **property** has its own topic and sends MQTT messages automatically (in JSON format) when updated. Therefore, the device must subscribe to these topics such as: "spaces/tfm-toy/collections/ ESP8266/things/thingID/properties/difficulty".

On top of that, the device is sending messages to that same topic to modify the properties each time a button/sensor changes value. And there is also a "…/data" topic that stores the historic registry of events for future analysis.

The following picture shows most of the properties as seen in the IoT platform in real-time:

Figure 7: Properties

There is a property for each button/led and sensor, and there is also an outputs array to send instructions when the device works on slave mode. The rest of the properties are variables to set the game mode, state of the device, difficulty, time limit…

The "Raw History" section with the timestamps can be viewed in the next figure.

Once created the "thing" with its properties, it is possible to read/write in its possible to access the values via HTTP/MQTT on the credentials page:

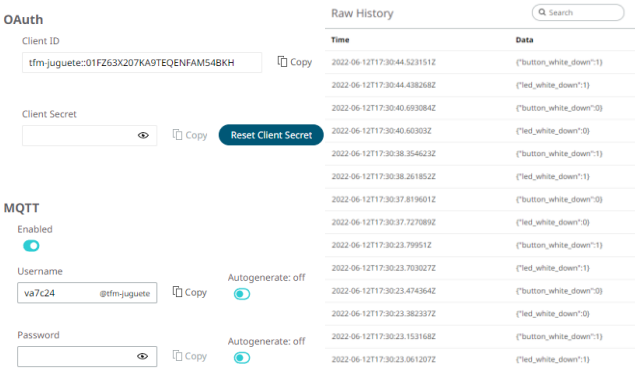

Figure 8: Credentials and Raw History

There are also other tools in the platform, such as an MQTT Inspector, API Inspector and log console to test the communication or a schema to modify/edit/clone the thing with its properties and configuration. But the main functionality of the platform comes from the **Functions**. Functions are divided into 2 parts: triggers and workers.

Triggers can be made with cron (timed) or activated via MQTT. For the project, the last one will be used, and each time the device indicates a change in the topic/property "state" the trigger will activate and call the worker.

Workers contain the main programming, with the code being in Python (although it can also be in Go). The worker programmed for the project is tasked with making an HTTP GET request to the platform API and accessing the "thing" properties. Once received the JSON and stored the values in variables, the worker proceeds to execute the state machine depending on the devices' state (Waiting for instructions, Running or Timeout when the game time reaches its limit). The worker contains the logic of the games to check the inputs (buttons and orientation) to be equal to the objectives of the game, either way, if correct or incorrect, the worker sends an HTTP POST message indicating the new state (Victory, Keep Trying or Timeout) and the next outputs' instructions for the device.

This allows for computing and programming in the platform instead of the device which brings a world of infinite scalability with a possible network of connected devices, instead of limiting the platform to a single device.

The other main capability of the platform is the Real Time Monitorization features. It allows the implementation of a dashboard to gather the information obtained from all devices and allow its study. For the project, a simple **dashboard** has been made to see the inputs, outputs and variables:

Figure 9: Dashboard

To implement the dashboard an **app** is required, which defines the credentials and access control. The apps can have different scopes depending on whether accessing the properties (thing) or raw history (data), as shown in the next image:

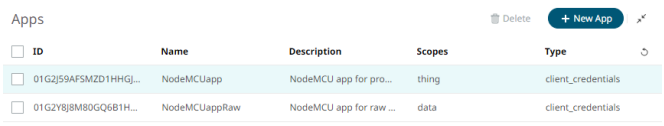

Figure 10: Apps and scopes

Then a **workbook** is made to connect the app and define the DataTable which is created automatically but requires to be modified with the format of each column for time/variable formats. Finally, multiple dashboards can be designed for one or multiple devices and tables/graphs showing the data gathered through the time played.

The data can also be exported to study or seen on the platform, but its main use is as an intake for possible workers in the future to adjust the difficulty according to the completion time and mistakes made in the game.

IV. ANALISIS, CONCLUSIONS AND FUTURE DEVELOPMENT

Firstly, regarding the **physical construction**, and as previously mentioned, the design is a prototype, and therefore it has room for improvement to make a scalable product. But it is a solid construction, perfect to test the combination of components and viability of the idea. As a result, it has been proven that basic components can make a very versatile toy with wide gameplay possibilities.

For future designs ergonomics must be improved as the size is too large and that requires selecting smaller buttons. Also, a more professional solution should be found for the PCB, headers and wiring soldering, as it requires too much time for each toy. And although it allows disconnecting easily the components, it would be better to use a custom PCB once planned the final design. And obviously, a rechargeable battery should also be included since the debugging capable will no longer be needed as states the next paragraph.

Secondly, although **programming** with the NodeMCU in PlatformIO has had its drawbacks and learning curve, once completed, it is important to mention that no further progress in the programming is needed, as the slave mode allows to follow the platform instructions and should require no further programming.

It could require a modification in case of a change of components or an interest in simplification of instructions (for example blinking the LEDs could be performed via a simple instruction instead of requiring the platform to turn them on and off periodically), but the functionality of the device can not be increased any further.

Also, the microcontroller's performance is a non-factor as most of the computing is done by the platform and the limitation are speed are due to lag and connection issues.

This means the rest of the programming will be done in the IoT platform. And it is, thirdly, the **IoT development** the part with the most room for future expansion.

Since the objective of the project was to test the device and communication with the platform, it has left the huge possibility of developing the entertaining platform:

- **New games** should be implemented. In the code annexed in the main document, it can be seen how easy it is to program the game state from the platform in Python, once all the communication and states have been set (10 lines of code instead of 100). And also, some ideas with also progressive difficulty are mentioned in the main document, but if introducing different components such as RGB LEDs, more puzzles can be created.
- The algorithm to change the **difficulty** and time limit based on the tracked **progression** can be made with a worker. Although it would be better implemented in combination with the next topic.
- **Analysis** and behavioural **studies** can be done once more devices have been made and a **network** of connected toys has been established, with Machine Learning regression studies of the cognitive development of the child for a better understanding of the logic they apply to solve a puzzle through the different stages of growth.

Last of all, the fourth point to mention to end the analysis of the project is also the **documentation** done in the main document, which explains the whole process making very simple the process of replicating it to add new toys to the network (both the construction of devices with the improvements mentioned and the easy cloning of the "thing" on the IoT platform) or simply to understand the different processes required to make a similar project (physical, programming or IoT).

During the development of the project **difficulties** appear and were either solved or avoided while maintaining the objectives in mind thanks to refocusing the problem. When it comes to wiring more preparation, investigation and planning of the connections would have avoided some time loss (for example the port expander was added after noticing the limit in the number of pins instead of preparing for it, or – as previously mentioned – the time spent on soldering instead of finding a better solution), but overall the construction ended with added functionality as it is possible to disconnect all components for testing or disassembling them to make a new one.

Similarly debugging mistakes proved tedious and time lost was increased due to the device having to load the corrected code and connect to both Wifi and the IoT platform each time a change was made, which takes more than a minute. But thanks to the switch from integrated **programming** to cloud-based, there was less need of changing the code. Another code has been made to simulate the workers-triggers in Visual Studio to avoid loading even the workers which also takes time.

This whole approach brought the idea of programming new games and processing the state machine online which not only simplifies it, but also facilitates the possibilities of scalability already mentioned.

Finally, in addition to the original objectives already met (prototype construction, validation, monitorization and IoT connexion), there have also been taking into consideration 3 SDGs:

- Education, a network of a hundred monitored devices could allow a deep analysis of the growth of infants' minds.
- Innovation, with the concept of having an entertaining and educative platform capable of programming and computing the toys online.
- Economic growth, a new market of connected toys with huge scalability and possibilities has been proven to be possible.