Diseñando un Fichador Personalizado
Designing a Customized Employee Time Tracker

Trabajo de Fin de Grado
Curso 2023–2024

Autor
Santiago Elias Rabbia

Director
Gonzalo Méndez Pozo

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Diseñando un Fichador Personalizado
Designing a Customized Employee Time Tracker

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Santiago Elias Rabbia

Director
Gonzalo Méndez Pozo

Convocatoria: Junio 2024

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Fecha publicación

Resumen

Diseñando un Fichador Personalizado

Cambios en las regulaciones modificaron el procedimiento por el cual los empleados fichan al entrar a sus trabajos en Acciona Facility Services, volviendo obsoletos muchos dispositivos existentes. Se necesitaba una respuesta rápida, por lo que se desarrolló un dispositivo de fichaje rentable que utiliza microcontroladores, un PCB personalizado y una caja impresa en 3D.

Este dispositivo IoT debe ser responsivo, intuitivo de usar y ofrecer un servicio casi ininterrumpido, enviando información a través de WiFi o datos celulares. El intercambio de datos se realiza por medio de API alojada en la nube, desarrollada usando Java Spring, que pueden ser montada como un contenedor de Docker, o como una Google Cloud Function.

Palabras clave

fichador, microcontrolador, pcb, impresión 3D, cloud API, IoT, programación asíncrona, Docker, GCP, Java Spring

Abstract

Designing a Customized Employee Time Tracker

Regulatory changes have prompted a shift in employee clock-in procedures at Acciona Facility Services, rendering many existing devices obsolete. A swift response was necessary, leading to the development of a cost-effective clocking device utilizing microcontrollers, a custom PCB, and a 3D-printed case.

This IoT device has to be responsive, intuitive to use and offer close-to uninterruped service, sending information through WiFi or cellular data. Data exchange is facilitated through cloud-hosted APIs developed using Java Spring, hostable as Docker containers or as Google Cloud Functions.

Keywords

clocking device, microcontroller, pcb, 3D print, cloud API, IoT, asynchronous programming, Docker, GCP, Java Spring

Contents

List of figures

Chapter 1

Introduction and objectives

Recent legislative changes introduced by both the European Union and the Spanish government have mandated alterations to employee clocking procedures. Notably, Spain now requires employees to clock in upon arrival at work, necessitating the implementation of reliable time-tracking systems. Additionally, the European Union has banned the use of clocking devices employing biometric identification methods such as fingerprint scanning or facial recognition.

Acciona Facility Services manages many thousands of employees throughout Spain, and it was of utmost importance to expeditely replace the non-compliant devices for different ones, as many companies were already being imposed hefty fines, from tens to hundreds of thousands of Euros, for not meeting the new regulatory requirements.

The company already had providers for many types of clocking devices, and many of them were compliant, but were priced in the hundreds of Euros each. Now, faced with having to replace them, the prospect of replacing *thousands* of units at considerable expense loomed large. Moreover, outsourcing these devices often meant committing to complex time-tracking ecosystems, adding further complications and increasing the complexity of the data the company manages.

These issues were brought to light to the innovation team at Acciona Facility Services, and were then tasked with bringing a solution to market.

1.1. Objectives

Given the preceding challenges and considerations, the development of a cost-effective device and the establishment of a cloud infrastructure were deemed necessary.

The objectives of this project are as follows:

- **Design and Prototyping**: Design a device utilizing microcontrollers and additional electronic modules, such as LCD displays and NFC readers, to

fulfill the specified requirements.

- **PCB Design**: Develop a Printed Circuit Board (PCB) layout to facilitate easy interconnection of the various electronic modules used in the device, ensuring efficient and reliable performance.
- **API Development**: Create an Application Programming Interface (API) capable of deployment as a Docker container or Google Cloud Function, enabling seamless communication between the device and cloud-based services.
- **Microcontroller Research**: Investigate the constraints and capabilities of microcontrollers, particularly RP2040-based microcontrollers, to inform design decisions and optimize performance.
- **3D Printing and Modeling**: Explore the fundamentals of 3D printing and 3D modeling necessary for designing a suitable enclosure for the device. Additionally, provide an overview and comparison of various materials suitable for enclosure fabrication, considering factors such as durability, cost, and aesthetic appeal.

Chapter 2

Breadboard Prototype

The first phase of developing the new clocking device involved thorough research into available components and microcontrollers in the market. Factors such as cost, functionality, and potential drawbacks were carefully evaluated to inform the selection process. Subsequently, selected components were tested by constructing a basic prototype on a breadboard to assess functionality and performance. Demonstrating the viability of the project at this stage was crucial for ensuring its continued development and success.

2.1. Hardware selection

Firstly, considering that the development was urgent, it would only be possible to use an already made controller. There were two routes to take:

- Using a **single-board computer**.
- Using a **microcontroller**.

The team's familiarity with *Raspberry Pi* products and their reputation for reliability led to the decision to utilize their offerings for the project. Given the lightweight processing requirements, the *Raspberry Pi Zero 2 W*, a cost-effective single-board computer with built-in WiFi capabilities, emerged as a suitable option. Alternatively, the *Raspberry Pi Pico W*, a microcontroller, presented another viable choice.

However, it's important to note that while the *Pi Zero* offers more features and functionality, it comes at a higher cost compared to the *Pi Pico*. In fact, it costs almost three times as much. Therefore, careful consideration is warranted to determine whether the additional expense justifies the benefits.

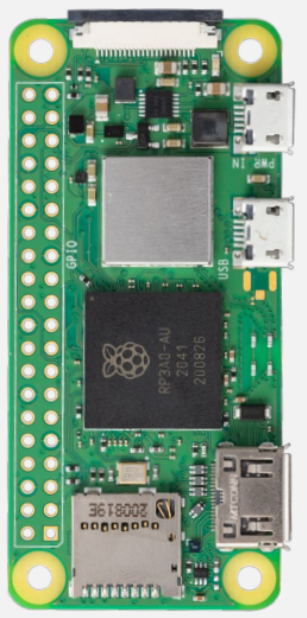

Figure 2.1: Raspberry Pi Zero 2 W

Figure 2.2: Raspberry Pi Pico WH

2.1.1. Single-board Computers

A single-board computer (*SBC*) is a complete computer built on a single circuit board. It integrates all the necessary components required for a functional computer system, including a central processing unit (CPU), memory (RAM), storage (usually in the form of a MicroSD card), input/output ports, and sometimes additional features such as networking capabilities (e.g., Ethernet or WiFi), audio/video output, GPIO (General Purpose Input/Output) pins for connecting external devices, and even USB ports.

SBCs are designed to be compact and efficient, which is the case of the *Pi Zero*, measuring about 65mm by 30mm while drawing about one Watt of power. Additionally, SBCs can run an operating system, such as Ubuntu.

This ability to run an entire operating system significantly enhances their versatility compared to microcontrollers out-of-the-box. Many peripheral devices can simply be plugged into a USB port and function seamlessly without requiring additional configuration. For instance, a 3G SIM card adapter, which will be necessary for future stages of the project, can be effortlessly integrated into the system, letting the operating system take charge of the communication with the mobile network, abstracting all these problems from the programmer.

2.1.2. Microcontrollers

A microcontroller is a compact integrated circuit (IC) that contains a central processing unit (CPU), memory (both volatile RAM and non-volatile flash memory), input/output peripherals (such as digital and analog I/O pins), and various other hardware components necessary for interfacing with external devices. Unlike single-board computers, microcontrollers are typically designed for specific tasks and embedded applications, often with real-time requirements.

One key characteristic of microcontrollers is their ability to execute dedicated firmware or software code stored in their internal memory. This code typically controls the behavior of the microcontroller, processes inputs from sensors or other external devices, and generates outputs to control actuators or display information.

The *Raspberry Pi Pico W* is a development board that utilizes the *RP2040* microcontroller chip. This board offers a range of features beyond its microcontroller, including onboard flash memory for program storage, versatile GPIO pins for interfacing with external devices, built-in USB connectivity for programming and power supply, and WiFi connectivity.

In comparison to single-board computers, the *Pi Pico* does not have an operating system. Instead, firmware can be loaded onto it, and it is this firmware that provides functionality to the board. This firmware allows the programmer to use programming languages such as C or MicroPython, which then control the microcontroller's behavior and interactions with external devices.

The absence of an operating system reduces the overhead associated with system management and resource allocation, resulting in faster boot times and improved reliability for time-critical tasks.

Taking into account our previously established requirements, which prioritize minimal points of failure, low computational demands, and cost-effectiveness, the logical preference leans towards the utilization of a microcontroller. For instance, single-board computers often rely on SD cards for storage, which can be prone to failure after prolonged use due to factors such as wear and tear or data corruption. In contrast, microcontrollers typically have simpler storage mechanisms, such as onboard flash memory, which are less susceptible to such issues. Thus, lower operating costs.

As a conclusion, a **microcontroller will be used**, and in particular, the ***Raspberry Pi Pico W***.

2.1.3. Electronic Modules

Now that the microcontroller has been selected, additional components are necessary to meet the project's requirements. Specifically, the device must integrate an NFC reader and an LCD screen. Additionally, other peripherals, such as a buzzer or LED, may also be evaluated.

We need to take into account the available buses and choose modules accordingly.

Relying solely on the datasheet alone is not enough to determine the buses that are usable concurrently, since if two different buses use the same pin, then they cannot be used simultaneously. Referring to the pinout diagram of the *Raspberry Pi Pico W* provided below, we can identify the available buses:

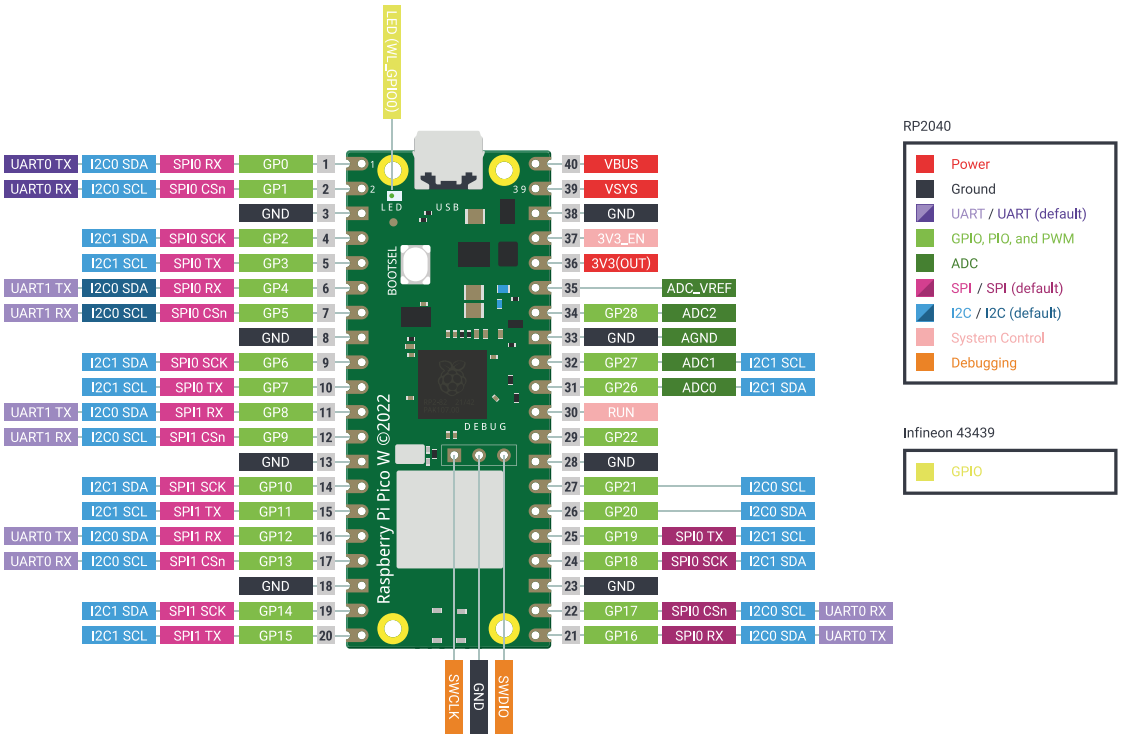

Figure 2.3: Raspberry Pi Pico W's Pinout

For example, each UART bus conflicts with I2C buses. Therefore, when selecting components, it's crucial to ensure compatibility with this layout [1].

NFC Module

Various NFC boards are available on the market. Ideally, the desired NFC board would offer extended range, compact form-factor, low power consumption, and support for multiple communication protocols such as UART, I2C, SPI, among others.

The *PN532* chipset produced by *NXP* offers all three types of buses mentioned before, that is: High Speed UART, I2C and SPI. Numerous boards utilizing this chipset are available on the market, with *ElecHouse*'s offering standing out for its excellent form-factor (about 4cm×4cm) and impressive range [4]. This specific board has been extensively replicated and is accessible at a low cost, and various providers offer open-source code for controlling it.

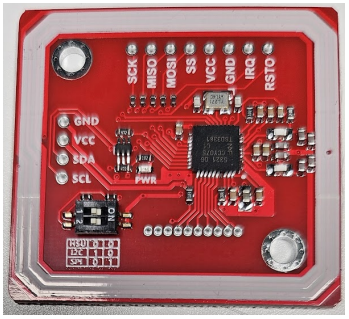

Figure 2.4: PN532 board

LCD Module

Initially, the idea of utilizing a black and white OLED display seemed appealing. However, as the size increased, so did the price, and even then, the displays were still too small. Consequently, the most practical decision was to adhere to LCD technology.

Upon researching LCD modules, the *LCD1602* module emerged as a suitable choice. It offers sufficient size, accommodating up to 16 characters per row across 2 rows, and provides satisfactory contrast. Additionally, it is available with I2C adapters for simplified control, requiring fewer pins on the microcontroller.

Figure 2.5: LCD1602 board

Buzzer

Beyond just relying on visual cues displayed on the screen to convey the device's status, it's essential to enhance the user experience by incorporating auditory feedback. When employees clock in to work, providing a distinct sound signal from a buzzer ensures they receive immediate confirmation of their action.

There are two types of buzzer:

- **Active buzzers**: they incorporate an internal oscillator circuit that generates the sound signal when voltage is applied. They are self-contained and do

not require external circuitry to produce sound. They are commonly used in applications where simplicity and ease of use are prioritized, as they can be directly connected to a power source to emit sound.

- **Passive buzzers**: they require an external oscillating circuit to produce sound. An alternating current is applied to create vibrations that produce sound waves. They offer more flexibility in sound frequency and intensity control but require additional circuitry for operation.

Due to the project's time constraints and the preference for simplicity, **active buzzers will be employed**. Additionally, since a single sound frequency suffices for the intended use, active buzzers are well-suited for the task.

Cellular Connectivity Module

In certain locations, access to WiFi networks may be limited or unavailable altogether. Even in areas where WiFi is accessible, signal strength may vary, leading to instances of poor connectivity. Consequently, it becomes imperative to incorporate cellular connectivity options to ensure internet access across diverse environments and conditions. By integrating cellular connectivity capabilities into the system, users can rely on alternative means of internet access, mitigating the limitations posed by WiFi availability and signal quality fluctuations.

Considering the time limitations, it would be benefitial to opt for a ready-made solution. However, it's worth noting that the Raspberry Pi Pico, being a microcontroller, cannot simply utilize a USB dongle with a SIM card adapter for cellular connectivity. A more intricate solution is necessary. Fortunately, there exists a module specifically designed for compatibility with the Pi Pico, developed by *Waveshare*. This module integrates the *SIM7020E* module, manufactured by *SIMCOM*, a prominent provider of wireless modules and electronics.

The *SIM7020E* module, known for its affordability and low power consumption, is specifically designed for NB-IoT[1] applications, making it an optimal choice for M2M[2] communication scenarios. Integrated into the *Waveshare* board, this module establishes communication with the Pi Pico through UART, using *AT*[3] commands for control and data exchange [8].

However, a significant tradeoff exists as this module only supports NarrowBand IoT. NB-IoT operates within the LTE (Long-Term Evolution) spectrum and is tailored for narrow bandwidths, making it ideal for low-power, wide-area IoT applications. While NB-IoT typically utilizes the 4G LTE spectrum, it operates at reduced data rates compared to conventional LTE, facilitating efficient communication with IoT devices while minimizing power consumption and costs.

[1] NB-IoT, short for Narrowband Internet of Things, is a low-power cellular technology designed for efficient communication between IoT devices and networks.

[2] M2M stands for Machine-to-Machine, and encompasses a wide range of applications where devices or machines communicate and exchange data without human intervention.

[3] AT commands are a standardized set of instructions used to communicate with and control modems and other serial devices.

Nonetheless, this limitation poses challenges as the range constraints of LTE networks persist, potentially leading to signal issues in underground facilities. In an ideal scenario, a module supporting 2G and NB-IoT would be preferable for the project. However, such modules are not readily available on the market for seamless integration with the Pi Pico. While *SIMCOM* does offer several variants of these modules with diverse capabilities, developing a custom solution would necessitate significant investment in terms of both time and resources.

However, the downside of this approach is that the module requires soldering of fourteen pins (the ones whose text is highlighted in white in Fig. 2.7), introducing an additional layer of complexity to the device assembly process. Furthermore, it is important to note the stacking headers featured in Figures 2.6 and 2.7, which consist of female headers with extended male pins. These pins are designed to penetrate the PCB and connect to another female header underneath. While this setup may appear standard, improper soldering, with excess tin reaching too high on the male part of the header, can lead to faulty connections with the PCB or breadboard. Therefore, if the team intends to assemble the device on-site, careful soldering is a must.

Figure 2.6: Waveshare's SIM7020E module, top view

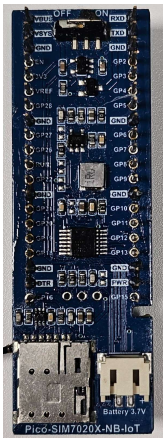

Figure 2.7: Waveshare's SIM7020E module, bottom view

Buses

As previously mentioned, various buses are available for interconnecting components. While the LCD is already set up for I2C, the NFC reader offers more flexibility in bus selection. However, to maintain simplicity, the NFC reader will also utilize the I2C bus.

I2C, short for Inter-Integrated Circuit, is a serial communication protocol commonly utilized for connecting microcontrollers and peripheral devices. It operates using two wires: *SDA* (Serial Data) and *SCL* (Serial Clock). Devices connected to the bus are addressed individually by unique addresses, and communication follows a master-slave configuration. The master device initiates communication and controls the bus, while one or more slave devices respond to commands. Data is trans-

ferred sequentially, with the master device generating clock pulses to synchronize communication [5].

However, I2C does have limitations. It operates at relatively low speeds compared to other protocols, which can impact performance in applications requiring high data transfer rates. The length of the I2C bus is also limited due to signal integrity issues, typically to a few meters, which may restrict the physical layout of devices in larger systems. In any case, none of these limitations affect this project's use case.

The *Pi Pico* features two I2C buses, providing an additional bus beyond what is required. It's important to mention that the two modules can share the same I2C bus since they have different I2C addresses. For instance, the PN532 always uses the *0x48* address, which cannot be modified. Consequently, two PN532 readers cannot share the same bus and must be placed separately. However, this is not an issue for the LCD as it uses a different address.

Note: While the PN532 produced by *NXP* does allow for the modification of the I2C address [6], it is conventionally maintained at *0x48* [7].

Matrix Keypad

A matrix keyboard offers an alternative means for employees to clock in or out by entering a numeric code, which can be particularly useful if an employee forgets or misplaces their card. The working principle of matrix keyboards is straightforward: as illustrated in Figure 2.9, they consist of intersecting rows and columns. Each key on the keyboard is positioned at the intersection of a row and a column. When a key is pressed, it creates a connection between a specific row and column, resulting in a unique electrical signal that can be interpreted by the microcontroller.

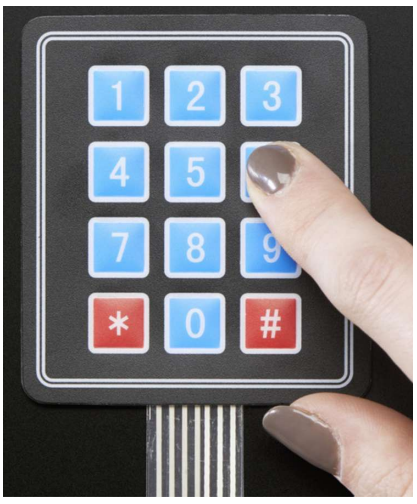

Figure 2.8: Matrix keypad

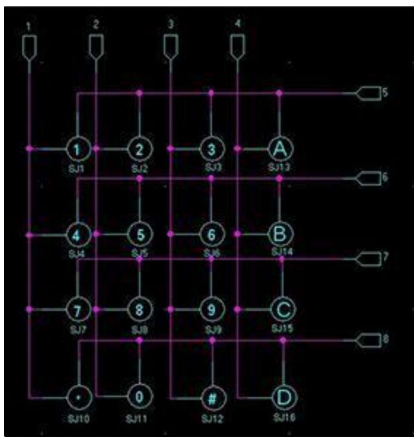

Figure 2.9: Matrix keypad's schematics

Matrix keyboards utilize a scanning technique to detect key presses. The microcontroller sequentially activates each row, while monitoring the columns for any signals. If a signal is detected, the microcontroller identifies the corresponding key based on the activated row and column, allowing it to register the key press.

One important consideration when using matrix keyboards is the potential for key ghosting or masking. Ghosting occurs when multiple keys are pressed simultaneously, causing the keyboard to register unintended key presses. Masking, on the other hand, happens when certain key combinations prevent other keys from registering. To mitigate these issues, careful design and debounce techniques must be employed to ensure reliable key detection.

During testing, it was observed that with sufficiently fast scanning and debouncing, the occurrence of ghosting or masking on the keyboard would likely not present a problem in practical usage scenarios. This assessment stems from the understanding that the likelihood of multiple key presses happening simultaneously on a wall-mounted device, such as the one being developed, is minimal.

Another challenge arises from the substantial use of GPIO pins on the microcontroller. The 4×4 keyboard depicted in Figure 2.8, consisting of 4 rows and 4 columns, requires 8 GPIO pins for operation. Considering that the Pi Pico offers a total of 29 GPIO pins, this allocation accounts for nearly 30 percent of available pins.

2.2. Prototype Assembly

Once the components have been selected and basic schematics drawn, the assembly process can commence. All components can be interconnected on a breadboard using jumper wires. It is important to highlight that the Raspberry Pi Pico WH comes with pre-soldered headers, facilitating easy integration for testing purposes. Only minimal soldering is required for certain components, such as the four pins on the NFC module and the fourteen pins for the Waveshare module.

Many of the components necessary for this prototyping phase can be obtained by purchasing a basic electronics kit, which typically includes a breadboard and jumper wires, and other basic components such as a matrix keyboard and buzzer.

In addition to the minor soldering required, the majority of components were effortlessly connected to the breadboard using jumper wires. The small footprint of the Pi Pico further simplified the connections to its pins.

In the subsequent sections, detailed explanations of the wiring configurations for all modules will be provided, offering insights into their respective functionalities and interconnections within the system.

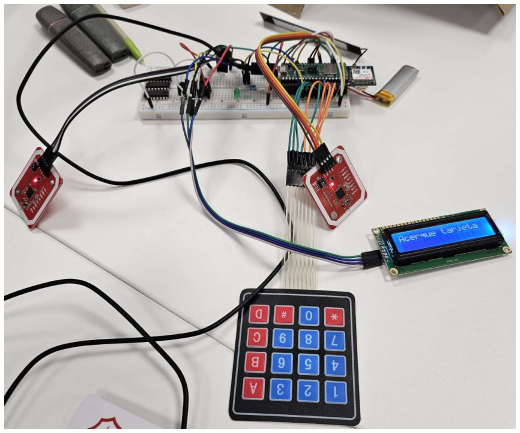

Figure 2.10: First prototype mounted on a breadboard

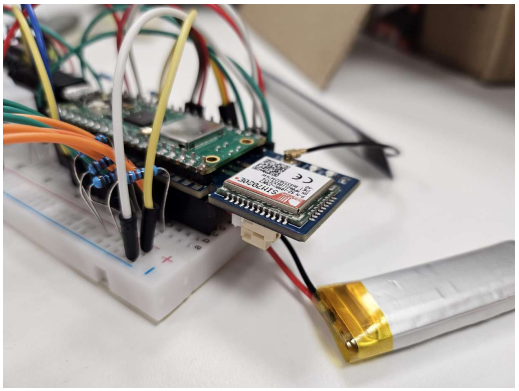

Figure 2.11: Zoom in on the Pi Pico and SIM7020E module

2.2.1. Matrix keypad wiring

Row Pins Configuration

- The 4 row pins are configured as INPUT with PULL-UP resistors enabled. This configuration allows the microcontroller to detect changes in voltage on these pins.
- Each row pin is connected to the positive supply voltage (3V) through a 10kΩ resistor. This ensures that when no button is pressed, the row pins maintain a high voltage level due to the PULL-UP resistors.
- When a button in a particular row is pressed, it creates a connection between the row and column pins, effectively pulling the voltage level of the corresponding row pin down to ground.

Column Pins Configuration

- The 4 column pins are configured as OUTPUT. During scanning, each column pin is individually set to false (logic low), one at a time, while the others remain at a high logic level (logic high or true).
- By setting a column pin to false, it creates a path to ground for the corresponding row pins. If a button in the corresponding column and row is pressed, the

voltage on the row pin is pulled low, indicating a button press.

Button Press Detection

- When a button is pressed, it forms a connection between a row and column pin. During scanning, when the corresponding column pin is set to false, it effectively pulls the voltage on the row pin down to ground.
- The microcontroller continuously scans each column pin, one by one, and if it detects changes in voltage on one row pin, then it already pinpointed the row-column pair, corresponding to a key press.

In summary, this configuration allows the microcontroller to detect button presses on the keypad by scanning each column and monitoring the voltage levels on the row pins. When a button is pressed, it creates a connection between a row and column pin, causing a change in voltage on the row pin, which is detected by the microcontroller.

2.2.2. Devices on the I2C bus

Thanks to the straightforward nature of I2C communication, connecting both the LCD display and the NFC reader requires just two wires each: one for the Serial Data Line (SDA) and the other for the Serial Clock Line (SCL), in addition to the necessary VCC and GND connections. With the Pi Pico offering two I2C buses, either or both buses can be used. This versatility is possible because the two devices possess distinct I2C addresses, as specified in their respective datasheets.

I2C Pull-up

Ensuring proper signal integrity is crucial for the reliable operation of I2C communication. One essential consideration in this regard is the implementation of pull-up resistors on both the SDA and SCL of the bus.

Pull-up resistors serve to maintain the default high logic level on the SDA and SCL lines when they are not actively being driven by the master or slave devices. Without pull-up resistors, the I2C lines may float, leading to undefined voltage levels and potential communication errors.

The pull-up resistors effectively “pull” the voltage level of the I2C lines to the high logic level when they are not actively being driven low by a device. This ensures clear signal transitions and prevents signal distortion or noise interference that could disrupt communication.

Typically, pull-up resistors are connected between the SDA and SCL lines and the positive supply voltage (VCC) of the system. The value of the pull-up resistors determines the strength of the pull-up effect and should be chosen carefully to balance signal integrity with power consumption.

During the initial stages of the project, pull-up resistors were manually added to the SDA and SCL of the I2C bus to ensure proper signal integrity. However, it was later discovered upon examination of the device schematics that both the LCD display and NFC reader modules already incorporated pull-up resistors on their respective boards, with voltage level translation from 5V to 3.3V.

This unintentional duplication of pull-up resistors led to a reduction in the overall resistance of the pull-up network and an increase in the strength of the pull-up effect on the I2C lines. The existing 4.7kΩ resistor in both the NFC reader and LCD modules inadvertently paralleled with an additional 10kΩ resistor, effectively reduced the overall resistance to approximately 3.2kΩ [3]. While this theoretically could result in faster signal transitions, it also heightened susceptibility to noise and interference, potentially leading to signal integrity issues. Additionally, the increased current draw from the additional pull-up resistors slightly elevated the power consumption of the system.

Despite these consequences, it is important to note that the unintentional duplication of pull-up resistors is unlikely to cause permanent damage to the devices involved.

Problems with the NFC reader

During a later phase of the project, an issue surfaced wherein the NFC reader would cease to respond after prolonged periods of operation. The only indication of this malfunction was rapid flickering of its red LED. Attempts to command the reader yielded no response, needing a power cycle to restore functionality. This could be achieved either by disconnecting and reconnecting the power source or by pulling down the "RSTPD_N" pin on the board. To mitigate this recurring problem, a solution involving the insertion of a transistor between the microcontroller and the VCC pin on the NFC board was devised. This approach was favored over direct soldering onto a minuscule 1-millimeter pin on the NFC board, which presented challenges and fitting problems.

2.2.3. The SIM7020E board

With the Waveshare board purposefully engineered to align with the pinout of the Pi Pico, no wiring is necessary beyond connecting the two components. This module interfaces with the microcontroller via UART for communication and utilizes additional GPIO pins for auxiliary functions such as power management.

A critical consideration is to avoid utilizing any pins already allocated by the SIM7020E board. This precaution can be ensured by referencing the datasheet and wiki documentation provided by the manufacturer [9].

Although the device includes a 3.7V Li-Po battery in the box, it was decided against its utilization due to concerns about introducing an additional point of failure to the device. Furthermore, its presence impedes the Raspberry Pi from undergoing a hard reset when disconnected from the power source.

2.2.4. Buzzer and other components

The active buzzer, requiring minimal setup, simply needs a transistor to toggle its state using a control signal from the microcontroller. It's essential to note that it must still be connected to a 5V source, as the 3.3V from the GPIO pins is insufficient, hence the requirement for the transistor.

Adding an LED to the project is similarly straightforward, as it can be controlled directly by a GPIO pin due to its lower voltage and current requirements. When paired with a 50Ω resistor, the LED emits sufficient brightness, even in well-lit conditions. However, ultimately, it was deemed unnecessary and excluded from the final design.

2.2.5. Considerations

The initial prototype incorporated two NFC modules, as can be seen on figure 2.10, intending to use one module for clocking in and the other for clocking out. This approach was adopted initially to streamline compatibility with a specific provider. However, this strategy was subsequently abandoned in favor of a simpler solution, whereby the worker's entry or exit status is determined through software processing.

Initially, the matrix keyboard was contemplated as an alternative method for clocking in or out by entering a numeric code. However, this approach was deemed impractical due to several concerns. Using the worker's national identification number as the input code would involve handling sensitive information, which was not ideal. Additionally, distributing individual codes to each worker would present logistical challenges. As a result, this approach was ultimately abandoned.

Chapter 3

PCB Design

Following the initial development on a breadboard, the natural progression in the design evolution was to transition to a perfboard assembly. This intermediate step not only served to condense the device's footprint but also offered a preliminary glimpse into its form and functionality. Subsequently, the culmination of this iterative process involved the design and fabrication of a PCB tailored to the project's specifications, marking a significant milestone in its realization.

3.1. Assembling on a Perfboard

Perfboards, short for perforated boards, are commonly used prototyping platforms in electronics. They consist of a board with a grid of holes spaced at regular intervals. These holes are surrounded by copper pads or traces that can be connected using solder to create electronic circuits. Perfboards allow electronic components, such as resistors, capacitors, integrated circuits, and other discrete components, to be soldered onto the board, facilitating the creation of temporary or semi-permanent circuits for testing and development purposes [2]. They serve as a practical and versatile tool for electronics hobbyists, engineers, and designers to quickly prototype and iterate on circuit designs before moving to more permanent solutions like printed circuit boards (PCBs). Examples can be seen in Figures 3.1 and 3.2.

In the context of this project, transitioning to a perfboard prototype served multiple crucial purposes. Firstly, it offered a tangible visualization of the future device's physical footprint, providing executives with a concrete representation of the project's direction and potential. This visual aid not only conveyed the scale and form of the device but also showed its feasibility and progress, instilling confidence in its development trajectory.

To commence this phase, a diagram detailing the placement of each component and its connection to the corresponding pins on the microcontroller was drafted. This diagram served as a blueprint, guiding the subsequent assembly process.

With the schematic as a roadmap, the assembly of the perfboard prototype

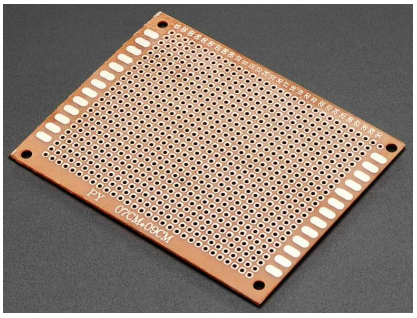

Figure 3.1: Back side of a perfboard

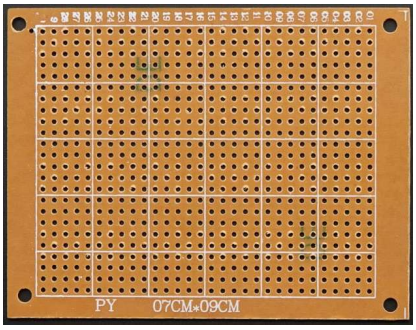

Figure 3.2: Front side of a perfboard

commenced. Components were transitioned from the breadboard to the perfboard, ensuring each element found its designated place. Careful consideration was given to the layout, optimizing spatial organization for efficient circuitry and minimal interference.

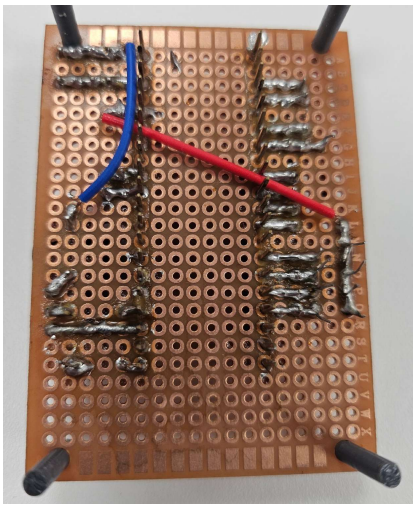

Figure 3.3: Back side of the assembled perfboard

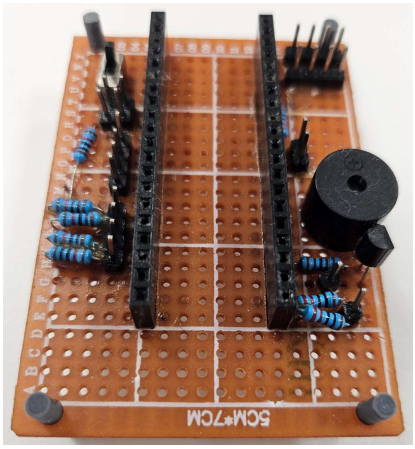

Figure 3.4: Front side of the assembled perfboard

3.2. Designing the PCB

Now that the perfboard provided a good idea of how the PCB could be, now it has to be properly designed. As it could be seen before, the perfboard's main job was to bring all components together, and that will also be the case for the PCB. It would give a much cleaner look than the perfboard, and be much more comfortable to plug components into, since on the perfboard traces could not overlap, limiting design possibilities. Now, with a multi-layer PCB, traces could go above and below each other, allowing for the placement continuous male headers for each component.

Having used the perfboard as a preliminary platform to consolidate components and understand the spatial dynamics of the circuit, the focus now shifts towards the meticulous design of the printed circuit board (PCB).

The primary function of the PCB mirrors that of the perfboard: to integrate and interconnect all components seamlessly. However, unlike the perfboard, the PCB offers advantages in terms of aesthetics, functionality, and ease of assembly.

Moreover, the transition to a multi-layer PCB introduces a realm of design possibilities previously unattainable with the perfboard. By using multiple layers, traces can now traverse both above and below the surface, allowing for more flexibility in component placement and routing. This feature allows for the implementation of continuous male headers for each component, which were not easily implemented with the perfboard.

Chapter 4

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Antes de la entrega de actas de cada convocatoria, en el plazo que se indica en el calendario de los trabajos de fin de grado, el estudiante entregará en el Campus Virtual la versión final de la memoria en PDF.

Bibliography

[1] Raspberry Pi Documentation. https://www.raspberrypi.com/documentation.

[2] DigiKey. How to use Perfboards. https://www.digikey.com/en/maker/blogs/2022/start-building-cleaner-perfboard-projects-using-these-simple-tips.

[3] ElecHouse. PN532 Board Schematics. . https://www.elechouse.com/elechouse/images/product/PN532_module_V3/PN532_shematic_drowing.pdf.

[4] ElecHouse. PN532 User Guide. . https://www.elechouse.com/elechouse/images/product/PN532_module_V3/PN532_%20Manual_V3.pdf.

[5] NXP. I2C Bus Specification. . https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[6] NXP. PN532 Specification. . https://www.nxp.com/docs/en/nxp/data-sheets/PN532_C1.pdf.

[7] Romkey, J. PN532 I2C Information. https://i2cdevices.org/devices/pn532.

[8] SIMCOM. SIM7020E Datasheet. https://files.waveshare.com/upload/e/e7/SIM7020E_SPEC_EN_190424.pdf.

[9] Waveshare. SIM7020E Board Wiki. https://www.waveshare.com/wiki/Pico-SIM7020E-NB-IoT.

Appendix A

Título del Apéndice A

Los apéndices son secciones al final del documento en las que se agrega texto con el objetivo de ampliar los contenidos del documento principal.

Este texto se puede encontrar en el fichero Cascaras/fin.tex. Si deseas eliminarlo, basta con comentar la línea correspondiente al final del fichero TFGTeXiS.tex.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes