Using repurposed TVBox for maker, embedded systems and IoT activities

Luiz Fernando Becher de Araujo
Western Parana State University - UNIOESTE
Cascavel, Brazil
luiz.araujo2@unioeste.br

Marcio Seiji Oyamada
Western Parana State University - UNIOESTE
Cascavel, Brazil
marcio.oyamada@unioeste.br

***Abstract*—The repurposing of illegal TVBox seized by the Brazilian Federal Revenue Service offers many opportunities for schools and universities. This type of initiative has a positive impact on the environment and the economy by avoiding the destruction of such devices, which generates waste and incurs costs. This work presents a methodology to convert these devices into prototyping board for maker, embedded systems, IoT, and robotics. This paper describes the process of identifying General Purpose Input/Output (GPIO) ports and presents practical examples of Python code for accessing them.**

***Keywords*—TVBox; repurpose; digital interface.**

I. Introduction

The use of prototyping boards allows the development of practical activities that can be used at different levels, from elementary school to higher education. These boards are characterized by the presence of GPIO (General Purpose Input/Output) ports that allow interaction with sensors and actuators.

The use of this type of solution in education enables the development of simple content on digital systems, such as blinking LEDs, through to more complex examples such as reading sensors, cameras and interfacing with actuators. Its use can range from primary to secondary and higher education to teach logic and programming. This type of board is also used for complex topics such as embedded systems, image processing, Internet of Things (IoT), robotics and edge computing.

There are several platforms on the market, each with its own characteristics and target applications. Arduino [1] is one of the most popular environments and is characterized by the presence of boards with different capabilities, a development environment and an extensive library of components that facilitate the development of applications. The Arduino framework is implemented in C++. Another widely used platform in this context is the Raspberry Pi [2]. In contrast to Arduino, Raspberry PI offers a system consisting of a multicore System-on-chip (SoC) and enables the execution of the Linux operating system and the development of graphical user interface (GUI) applications. The Raspberry PI enables the use of a wide range of programming languages such as C, C++, JavaScript and Python. Another well-known solution is the ESP8266/ESP32 family from Espressif [3]. This solution is characterized by its low cost and the presence of Bluetooth and WIFI interfaces, which facilitates the development of IoT solutions. Software development on this platform can be carried out using the Arduino or ESP-IDF framework.

TVBoxes are equipped with specialized hardware for media decoding, support both wired and wireless connectivity, and often include USB ports for peripheral integration. Manufacturers such as Rockchip and Amlogic produce Systems-on-Chip (SoCs), specifically designed for these devices based on ARM processors, integrating specialized hardware components for multimedia processing, such as video decoders.

This paper introduces a methodology to repurpose illegal TVBoxes for use in GPIO-based projects. The main point is the methodology to detect available GPIO ports that enable the connection of sensors and actuators. These illegal TVBoxes, which are confiscated by the Brazilian Federal Revenue Service, would be destined for destruction, resulting in waste and cost. Therefore, this work aims to find a more suitable destination for these types of devices, resulting in an improved low-cost teaching and learning solution, avoiding the environment and cost impact.

II. TVBox

The brazilian Federal Revenue Service seizes several models of TVBoxes that are being sold illegally in Brazil. These devices commonly referred to as IPTV, allow the consumption of audiovisual content transmitted over the Internet using the TCP/IP protocol stack, instead of traditional methods such as cable or satellite [4]. This requires specialized hardware to decode different media formats and support for wired and wireless internet connections. Several companies have specific SoC for this type of product, such as the platforms from Rockchip and Amlogic [5] [6].

Some types of TVBoxes are legal and can be used as they allow access to official streaming applications such as Netflix, HBO and others. Illegal IPTV refers to devices that allow users to access a list of channels that they have obtained for free or rented at a lower price, as they illegally obtain the content of the paid channels over the Internet. This type of solution is considered illegal in many countries and constitutes digital content piracy [7]. In Brazil, piracy is treated as an infringement of copyright, which is classified under Art. 184 of the Criminal Code [8]. The piracy problem can also be present in legal IPTV devices, since illegal apps can be installed to access the content. Another concern raised by Anatel (Brazilian National Telecommunications Agency) is about cybersecurity risks posed by illegal devices, including data theft and potential threats to telecommunications networks [9].

The confiscated illegal TVBoxes are destined for destruction by the Brazil Federal Revenue Service, which results in electronic waste and has a negative impact both financially and environmentally. For this reason, the Federal Revenue Service and its regional offices have started donating them to universities and encouraged the process of repurposing and converting these devices into minicomputers. When a TVBox is repurposed, the original operating system (usually a modified version of Android), which often contains software for pirating audiovisual content, is removed and replaced with a Linux distribution to reuse the hardware.

III. Related works

Several TVBox reuse campaigns are underway. The typical use for repurposed TVBoxes is as minicomputers in an educational context [10]. The project *Além do Horizonte - Beyond the Horizon*, developed by IFSul - Minas Gerais - Brazil, has already distributed TVBoxes to more than 64 schools [11]. Sobrinho et al. [12] propose the use of repurposed TVBoxes as thin clients in educational environments to improve the limited performance of this type of devices.

Another possibility is the use of TVBoxes as IoT devices. Sato et al. [13] uses repurposed TVBoxes in a people counting system that uses lightweight image recognition algorithms for edge processing. In [14], the authors show the advantages of repurposed TVBox over commercial solutions in terms of carbon footprint using the Brazilian energy matrix.

This work differs from the other works as it converts TVBoxes for use in maker activities, embedded systems and IoT, providing a solution similar to a Raspberry PI board. Other proposals trying to use the TVBox for the same purpose, needs an additional board (e.g. Arduino) to access the sensors and actuators. Our solution eliminates that need, enhancing environmental, financial, and educational value.

IV. Methodology

This section presents the methodology used in the repurpose of the TVBox and the search process of GPIOs in the board.

A. Repurpose

As part of the agreement with the brazilian Federal Revenue Service, some TVBox models were obtained. To reuse them, the Armbian operating system is usually the first choice, as it is a distribution optimized for ARM devices, based on Debian or Ubuntu, and for which there are some community initiatives to port the distribution to some TVBox models [15].

Among the TVBoxes sent by the Federal Revenue Service, the *in X Plus* model was selected (Table I), because it met the criteria of being able to install Armbian on the internal EMMC of the TVBox and thus ensure the replacement of the illegal software. The *in X Plus* is a TVBox with a relatively old SoC that still runs the 32-bit version of the ARM processor with the Rockchip RK3228 SoC.

Table I
TVBox configuration used in this work

SoC	Rockchip RK3228A
CPU	4x Cortex-A7 1,2 GHz
GPU	Mali-400
RAM	2 GB
Storage	8 GB eMMC

The board contains multiple unpopulated pins, increasing the likelihood of accessible GPIOs, since schematics for these devices are rarely available. Figure 1 shows the board used in this work. In the upper part, one can see several places where the design of the board offers space for the installation of integrated circuits that were not used in this device.

B. GPIO discovery

Due to the lack of a schematic of the board, which is usual for this type of product, the search for the input and output pins must be done. Based on the work of [16], a script was written to set a port to output mode and alternately write 0 and 1. Using a circuit with an LED and a resistor, each pin on the board is tested with a probe, and when the LED flashes, the number of the port and its physical position on the board is determined. The Listing 1 presents the sequence of commands for testing port 32.

After the Linux installation on the *in X Plus* board, the directory /sys/class/gpio showed the existence of 4 GPIO controllers with 32 ports each, for a total of 128 I/O ports. It is important to note that these ports are used for a variety of functions, e.g. for accessing the main memory, eMMC, HDMI, Ethernet, WIFI and others. Analysis of the SoC datasheet facilitates

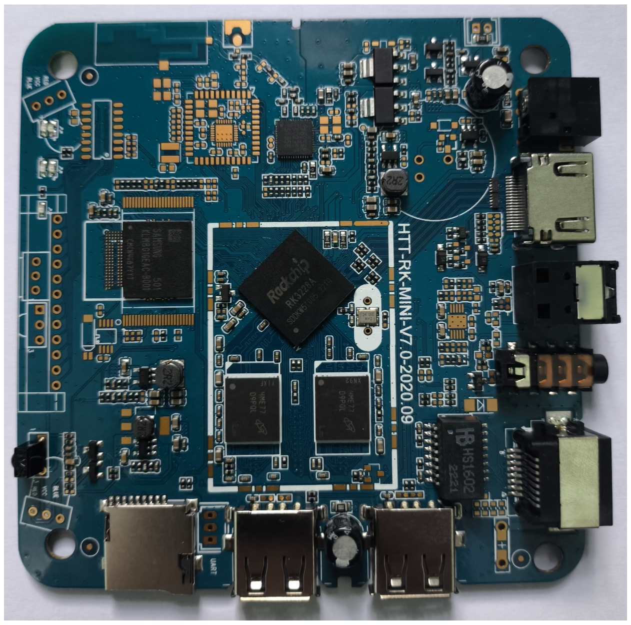

Fig. 1. Repurposed TVBox board. Model: *in X Plus*

the exclusion of ports already committed to these essential functions.

```
echo 32 > /sys/class/gpio/export
cd /sys/class/gpio/gpio32
echo 1 > /sys/class/gpio/gpio32/value
sleep 1
echo 0 > /sys/class/gpio/gpio32/value
```

Listing 1. Code for search GPIO pins

The Linux GPIO communication subsystem uses a pin nomenclature from 0 to 127, while the SoC datasheet uses the format $GPIO(X1)\_(X2)(X3)$. The equation 1 is used for the conversion, where the value of $X2$ can be $A = 0, B = 1$ or $C = 2$. For instance the port GPIO0_A1, is equivalent to the port 1, $GPIO = 32 * 0 + 8 * 0 + 1$.

$$GPIO = 32 * (X_1) + 8 * (X_2) + X_3 \tag{1}$$

Usually, TVBox devices have a serial port used to debug and monitor the boot messages. When starting to find GPIOs, the RX and TX pins is the first option to start the discovering process. In this work, only the datasheet for the RK3229 version was found on the internet, which the difference is only the processor with a higher frequency. After analyzing the datasheet and also the physical design of the board, some ports were mapped directly. For example, the RX and TX pins on the board were mapped to pins 41 (GPIO1_B1/UART1_TX) and 42 (GPIO1_B2/UART1_RX). The equation 1 is used to discover the pin number. For example, GPIO1_B1 is mapped to pin 41 ($GPIO = 32*1+1*8+1 = 41$). Figure 2 shows the position of the pins on the board in detail.

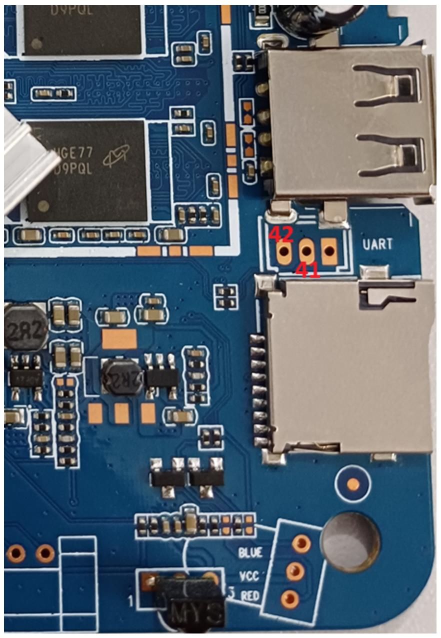

Fig. 2. Location of the RX (41) and TX (42) pins

I2C is a two-wire serial communication protocol consisting of the *serial data line (SDA)* and the *serial clock line (SCL)* [17]. The datasheet shows that the RK3228 SoC has four I2C controllers. This type of interface is very interesting in the context of embedded and IoT applications, as there are a large number of sensors and actuators that use this interface for communication.

One of the I2C controllers is used by the HDMI interface, which is responsible for detecting video modes on connected TVs and monitors. To avoid conflicts, it was necessary to identify alternative I2C pins available on the board. These pins will be usually available in parts of the board without soldered CIs. The search revealed that the I2C0 controller pins, specifically *GPIO0_A0= I2C0_SCL* and *GPIO0_A1= I2C0_SDA*, are accessible. It is important to note that this configuration is specific to this board, on which the designers have planned to include additional I2C peripherals. Figure 3 illustrates the physical location of the I2C pins on the board. In addition, 3.3V and GND pins were discovered, which are used to power the peripherals.

The search process was continued to try to map more GPIO pins on the board. The process was successful in mapping the pins associated with MMC1. The RK3228 has two MMC

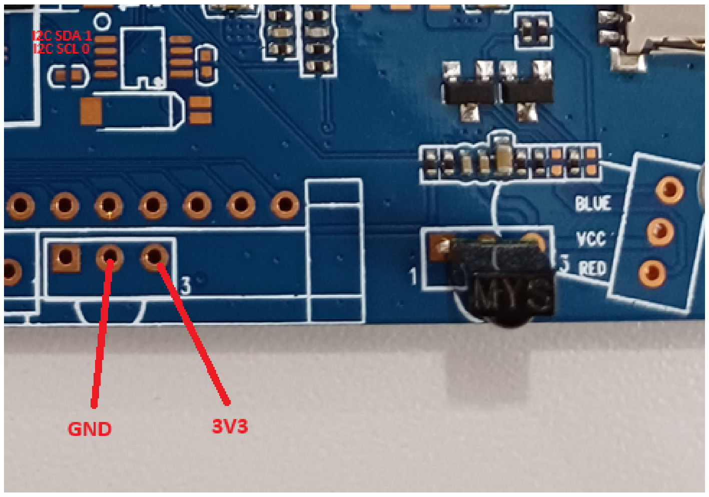

Fig. 3. Location of the I2C pins

controllers, one of which is used by the board's SD card slot. The second is physically mapped on the board, but without an MMC chip soldered on. It is important to note that the operating system and all files are stored on the eMMC (embedded MMC), which is soldered onto the board and uses a different set of pins. Figure 4 shows the detected MMC pins and their physical location on the board.

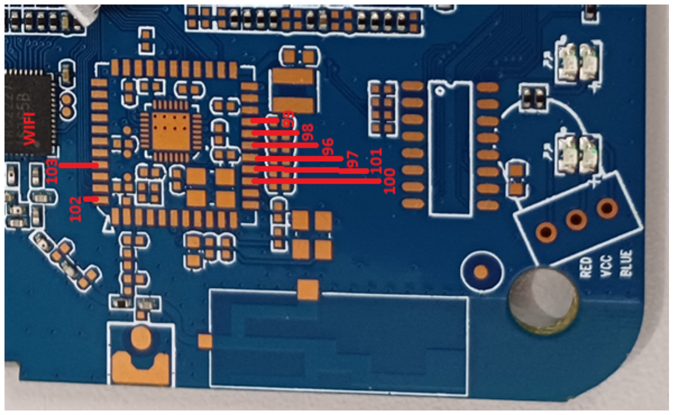

Fig. 4. MMC pins location

The table II shows all pins that were mapped. After completing this process, it was possible to identify the pins of the I2C interface and 10 GPIO pins. To facilitate the access to the ports, it was found that a number of holes on the board were not used. We have decided to solder pins in these holes, making it easier to assemble circuits with wires and protoboards. The connection between the port on the board and the pin was

Table II
DISCOVERED PINS

GPIO	Description
0	GPIO0_A0/I2C0_SCL
1	GPIO0_A1/I2C0_SDA
41	GPIO1_B1/UART1_TX/UART2_TX
42	GPIO1_B2/UART1_RX/UART2_RX
96	GPIO3_A0/SDMMC1_CLKO
97	GPIO3_A1/SDMMC1_CMD
98	GPIO3_A2/SDMMC1_D0
99	GPIO3_A3/SDMMC1_D1
100	GPIO3_A4/SDMMC1_D2
101	GPIO3_A4/SDMMC1_D2
102	GPIO3_A6/UART1_RTSN
103	GPIO3_A7/UART1_CTSN

made with enameled wires, which are often used as jumpers on boards to correct defective tracks or connections. To prevent the soldering process from becoming too complex and time-consuming, only a few pins were soldered. The final version is shown in Figure 5.

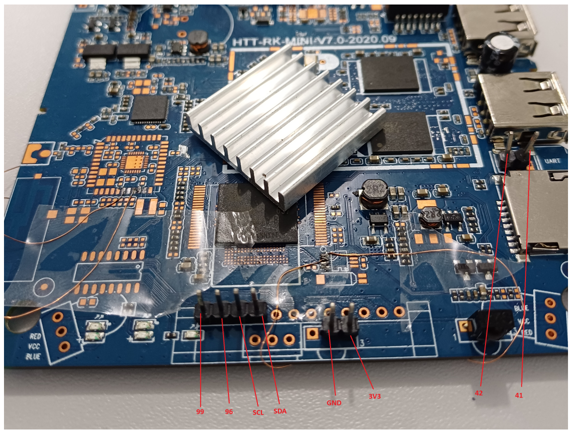

Fig. 5. Final GPIO pins

V. RESULTS

Following the mapping process, a series of practical activities have been developed to facilitate the use of the board.

Since the TVBox uses the Linux operating system, alternatives for the use of GPIO in this system were explored for the software development. The solution selected is the Blinka [18] library, which uses the CircuitPython API to access GPIO ports. This library is very popular in the Raspberry community and has a variety of components that facilitate communication with sensors and actuators.

The process of installing the libraries is described on the Github page of this project [1] , which contains all the necessary steps to install the dependencies and the library.

A. GPIO access

One of the first practices using prototyping boards is the blinking of the LEDs. In this practical example, GPIO port 42 was used to write the value to the LED. The physical connection follows the connection scheme: $GPIO42$ - LED - $RESISTOR$ 220Ω - GND. The code in the Listing 2 is used to make the LED blink. At the beginning of the code, an explicit mapping of which board will be used is necessary. Since the RK3228 is not officially supported by Blinka, to enable the use of the Blinka library, the RK3328 was used as the target board (lines 2 and 3 of the code), which also features 128 GPIO ports, enabling access to all required ports. Blinka uses the mapping $(controller, port)$ to define which pin is used. As each controller has 32 ports, the definition of port 42 is given by the value Pin((1,10)), Controller 1, Port 10. After initialization and configuration as an output port, the state of the LED is changed by simply writing the desired output to the value attribute of the port.

```
1  import os
2  os.environ["BLINKA_FORCEBOARD"]="ROC-RK3328-CC"
3  os.environ["BLINKA_FORCECHIP"]="RK3328"
4  import time
5  import board
6  import digitalio
7  from adafruit_blinka.microcontroller.generic_linux.
       libgpiod_pin import Pin
8
9  pin = Pin((1,10))  # (x,y) = 32*x + y= 32*1 + 10 = 42
       (GPIO 42)
10 led = digitalio.DigitalInOut(pin)
11 led.direction = digitalio.Direction.OUTPUT
12 while True:
13     led.value = True
14     time.sleep(0.5)
15     led.value = False
16     time.sleep(0.5)
```

Listing 2. Blink LED

For the use of buttons, the Blinka library provides the keypad component that manages all signal handling (reading and debouncing) and makes the events available to the application. The code 3 presents the processing events of a button connected to port 99. It is interesting to note that some ports of the RK3228 SoC have built-in pull-up resistors, which makes it easier to connect buttons to the port.

```
button=Pin((3,3))
keys = keypad.Keys((button,), value_when_pressed=
    False, pull=True)
while True:
    event = keys.events.get()
    if event:
        print(event)
```

Listing 3. Button event handling

B. I2C connection

Two components were used for the I2C connectivity tests: a 64x128 pixel SSD1306 OLED display and a BMP280 temperature and pressure sensor. The I2C protocol allows multiple components to be connected to the same bus, with the devices being differentiated by their address. In this example, the display is mapped to address 0x3C and the BMP280 is mapped to address 0x76. The interesting part of using Blinka and CircuitPython is the availability of libraries. In this case study, both the display and the BMP280 have libraries that only need to be installed. The code 4 shows the initialization of the I2C controller on ports 0 and 1 discovered in the mapping process, as well as the initialization of the display and the BMP280 sensor.

```
i2c = busio.I2C(Pin((0,0)), Pin((0,1)))
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c, 0
    x76)
```

Listing 4. I2C initialization

Figure 6 presents the circuit with the BMP280 sensor and the OLED display, which shows the values for temperature, pressure and altitude. As the TVBox has an HDMI output and an Ethernet connection, there are numerous possibilities for variations on this practical activity, for example creating a graphical user interface to display the results. Another option is to send the data to the internet, as it is common in IoT solutions. In Figure 7, the code has been modified to send the sensors data obtained from the BMP280 to the ThingSpeak platform, enabling online access.

C. Case study

The board developed in this work was used in a case study in an elementary school in the city of Cascavel, PR, Brazil. The school develops some extracurricular logic activities using Arduino. Figure 8 shows a circuit developed by students aged 12 to 13, aimed at simulating a queue management system. The system uses two buttons to increment and decrement the counter. Each time the button is pressed, the system turns on an LED, emits a beep and the new ticket value is shown on the OLED display.

The board developed in this work was used to redesign the solution as the students wanted the ticket to appear on a TV instead of a small OLED display. It fully met the requirements thanks to its native HDMI output and GPIO pins. Therefore, the physical circuit using the buttons developed in the Arduino

[1] https://github.com/msoyamada/XPlusGPIO/

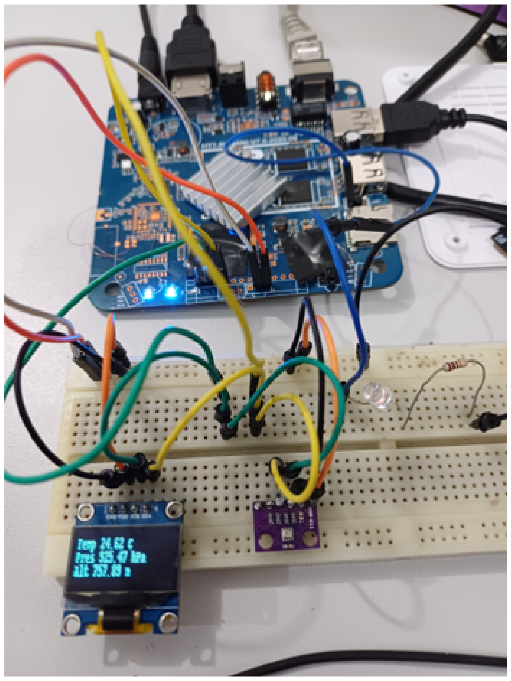

Fig. 6. Display e BMP280 access using I2C

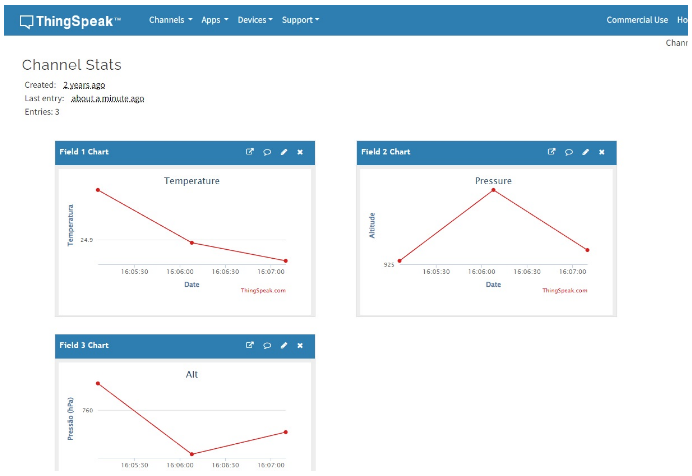

Fig. 7. Data sent to Thingspeak platform

version was reused without modification. The circuit with the repurposed TVBox is shown in Figure 9.

The software was rewritten since the Arduino version uses the C++ language. In the TVBox version, the Python language was used and the button handling was performed with the code shown in the subsection V-A. To display the value on the TV,

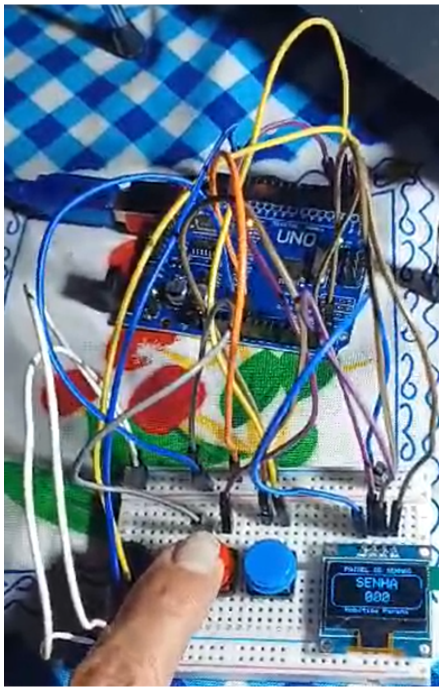

Fig. 8. Original version of queue management system using Arduino

the PySimpleGUI library was used [19] to create a full-screen graphical interface. Another feature implemented in this case study was the alert sound to inform a new ticket number. The playsound package available in Python was used to implement this feature. When comparing the original solution in Arduino with the one developed using the TVBox, it is possible to notice that all the resources requested by the students, such as the TV output and sound, which would have been difficult to implement with the Arduino, were easily implemented with the board developed in this work.

Figure 10 shows the students presenting the developed project at school during the break to organize the queue for the school snack.

VI. Conclusion and Future Works

The repurpose of illegal TVBoxes in Brazil is a very interesting initiative from an environmental, financial and educational point of view. The possibility of using these devices, that would otherwise be destroyed, for educational purposes holds considerable potential for transformation.

This article presents another way of using repurposed TVBoxes for use in the fields of digital systems, embedded systems, IoT, maker and robotics. Considering the resources provided by the TVBox, such as USB ports allowing the

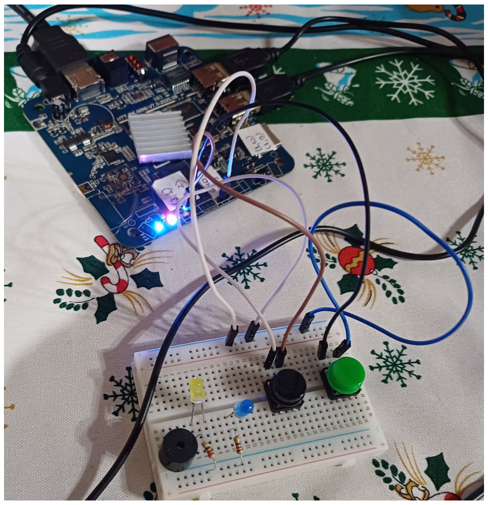

Fig. 9. Queue management system using the TVBox

Fig. 10. Project presentation

connection of keyboard and mouse, HDMI, GPIO and network, its use is extremely flexible and can be applied in different contexts. Considering these resources, the solution presented in this work is equivalent to available in the Raspberry PI 2 and 3 commercial boards. In the context of Brazilian and many other countries schools and universities, the cost of these commercial solutions often makes their use prohibitive. Reusing TVBoxes that would otherwise be destroyed is therefore a cost-effective solution. In addition, the use of Blinka and CircuitPython, libraries that are widely used on other boards such as the Raspberry to access GPIO, makes the solution generic and easy to adopt. It is important to emphasize that no changes to the library code were necessary.

Future work includes improving the Blinka library for automatic platform detection so that the explicit declaration of the SoC in the code will no longer required. In addition, practical activities focusing on robotics, such as motor and servo control will be developed to explore the board's capabilities. In addition, a comparison in terms of performance and energy consumption between the repurposed TVBox and the commercial platform such as Raspberry will be performed to determine the differences and limitations of the solution.

Acknowledgment

Thanks to Brazilian Federal Revenue Service. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior –Brasil (CAPES) – Finance Code 001

References

[1] Arduino. (2025) Arduino hardware. [Online]. Available: https://www.arduino.cc
[2] Raspberry. (2025) Raspberry PI. [Online]. Available: https://www.raspberrypi.com
[3] Espressif. (2025) Esp32. [Online]. Available: https://www.espressif.com
[4] S. Zeadally, H. Moustafa, and F. Siddiqui, "Internet protocol television (IPTV): architecture, trends, and challenges," *IEEE Systems Journal*, vol. 5, no. 4, pp. 518–527, 2011.
[5] Rockchip. (2015) Rockchip rk3229 datasheet. Rockchip. [Online]. Available: https://rockchip.fr/RK3229datasheetV1.2.pdf
[6] Amlogic. (2020) S905x3 datasheet. [Online]. Available: https://datasheet4u.com/datasheet-pdf/Amlogic/S905X3/pdf.php?id=1483651
[7] P. A. F. Neto and M. S. Silva, "Direitos autorais e internet: o streaming ilegal de obras audiovisuais," *Cadernos de Prospecção*, vol. 12, no. 5, pp. 1190–1190, 2019.
[8] Brasil. (2003) Lei nº 10.695, de 1º de julho de 2003. [Online]. Available: https://www.planalto.gov.br/ccivil_03/LEIS/2003/L10.695.htm
[9] R. Peret. (2023) Novas punições para tv pirata com multa até para os usuários. [Online]. Available: https://tribunaonline.com.br/economia/novas-punicoes-para-tv-pirata-com-multa-ate-para-os-usuarios-145459
[10] P. J. Varela, M. S. Santos, E. Endres, M. D. Leite, M. Albonico, M. Tollemache, and S. R. Felicio, "Projeto transformar: Conversão de equipamentos eletrônicos ilegais (iptv's) em minicomputadores para enfrentar as desigualdades educacionais em escolas públicas," in *Anais do XXIX Workshop de Informática na Escola*. SBC, 2023, pp. 1171–1181.
[11] D. d. Cunha Mendes and F. Corsini, "O impacto social do projeto além do horizonte nas escolas públicas municipais beneficiadas," *Jornada Científica e Tecnológica*, vol. 15, no. 3, 2023.
[12] M. P. Sobrinho, J. Sousa, F. Carvalho, C. Silva, and L. Santos, "Explorando tv boxes em ambientes educacionais: Vantagens e limitações operando como thin client," in *Anais da XII Escola Regional de Computação do Ceará, Maranhão e Piauí*. Porto Alegre, RS, Brasil: SBC, 2024, pp. 249–258. [Online]. Available: https://sol.sbc.org.br/index.php/ercemapi/article/view/30192
[13] G. Sato, G. Luz, L. Gonzalez, and J. Borin, "Reaproveitamento de tv boxes para aplicação de contagem de pessoas na borda em cidades inteligentes," in *Anais do VIII Workshop de Computação Urbana*. Porto Alegre, RS, Brasil: SBC, 2024, pp. 197–209. [Online]. Available: https://sol.sbc.org.br/index.php/courb/article/view/30000

[14] G. P.C.P. da Luz, G. Sato, L. Gómez Gonzalez, and J. Borin. (2024, 07) Repurposing of tv boxes for a circular economy in smart cities applications repurposing of tv boxes for a circular economy in smart cities applications. [Online]. Available: doi://10.21203/rs.3.rs-4619533/v1
[15] Armbian. (2024) Armbian. Armbian. [Online]. Available: https://www.armbian.com/
[16] M. M. Aguirre and C. A. Fleury, "Identificando e utilizando portas GPIO da TV-BOX descaracterizada," Instituto Federal de Goias, Tech. Rep., 2023.
[17] J. Wu. (2022) A basic guide to I2C. [Online]. Available: https://www.ti.com/lit/an/sbaa565/sbaa565.pdf
[18] Blinka. (2025) CircuitPython blinka library. [Online]. Available: https://circuitpython.org/blinka
[19] PySimpleGUI. (2025) PySimpleGUI. [Online]. Available: https://www.pysimplegui.com/