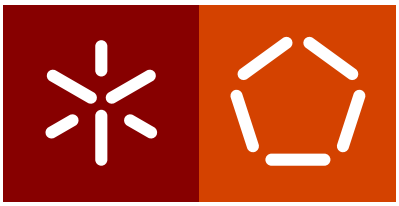

Universidade do Minho
Escola de Engenharia
Departamento de Engenharia Eletrónica
Embedded systems

Assignment 1: Report

Development of a 3 button TV remote control — Analysis & Design

José Pires A50178
Hugo Freitas A88258

Supervised by:
Professor Tiago Gomes

October 19, 2021

Contents

List of Figures

List of Listings

1 Introduction

The present work illustrates the application of the two first stages of the Waterfall methodology — Analysis and Design — to develop a Television (TV) remote control. This type of project begins with the establishment of a contract between the client (Samsung company) and the project team, clearly defining the problem statement and deriving the product requirements and constraints associated to the project. It should be noted, however, that both roles are played out by the authors. A market research is performed to gain more insight over this market and the product placement, and the product overall characteristics.

In the analysis phase, the product requirements are derived — defining the client expectations for the product — as well as the project constraints — what the environments limits about the product. Finally, the theoretical foundations are outlined, providing the basic technical knowledge to undertake the project.

In the design phase, the product development starts, specifying the system in terms of hardware and software and its associated interfaces, the error handling required, and the design verification.

1.1 Problem statement

The first step of the project is to clearly define the problem, as a result of the contract established between the client and the project team, yielding, in this case, the following project statement:

"Design a remote control with three buttons that can remotely control the television (TV). It should be very light, powered by batteries and controls your TV via an infrared emitter. The TV has a built-in infrared receiver. A button on the remote control switches the TV on/off and will be labeled with the word "Power". The other two buttons are used to scroll up/down and select the available channels and they are labeled with the arrows up/down."

1.2 Market research

A TV remote is a device which is used to operate a television from distance in a wireless mode. It also makes the TV usage simpler, more user friendly with its suggestive buttons. These buttons control functions such as power, volume, channel switch and various other features.

TV remotes are composed by the TV remote Shell, the TV remote membrane, one LED and a data acquisition & Infrared emitter PCB, as illustrated in Fig. 1.1. The unit cost of universal TV remotes is about 3 to 5 EUR.

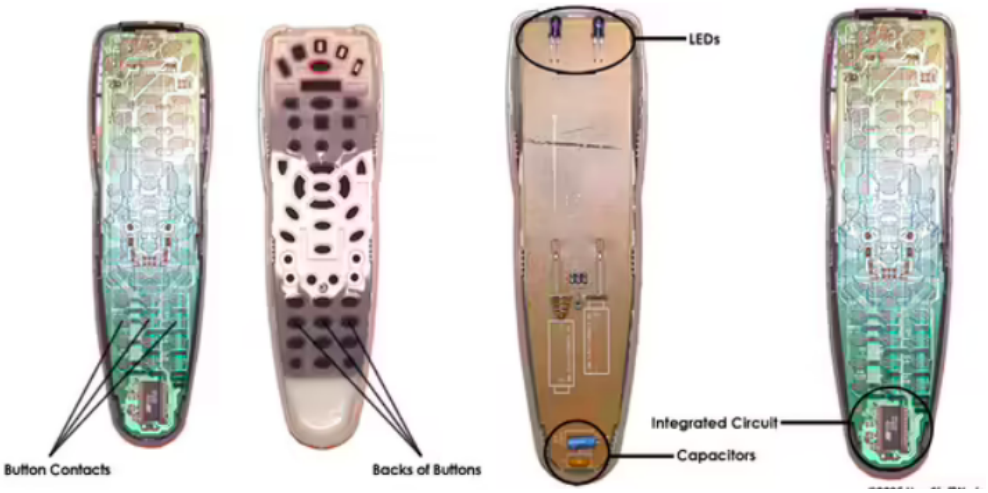

Figure 1.1: TV Remote control bill of materials, withdrawn from [1]

As can be seen in Fig. 1.2, the amount of televisions sold per year is about 200 million per year, with a tendency to increase over the next years. Thus, at least the same amount of TV remotes sells is expected, as each new TV requires one remote control, but it is expected to be exceeded due to TV remote replacement arising from its malfunctioning or bad usage.

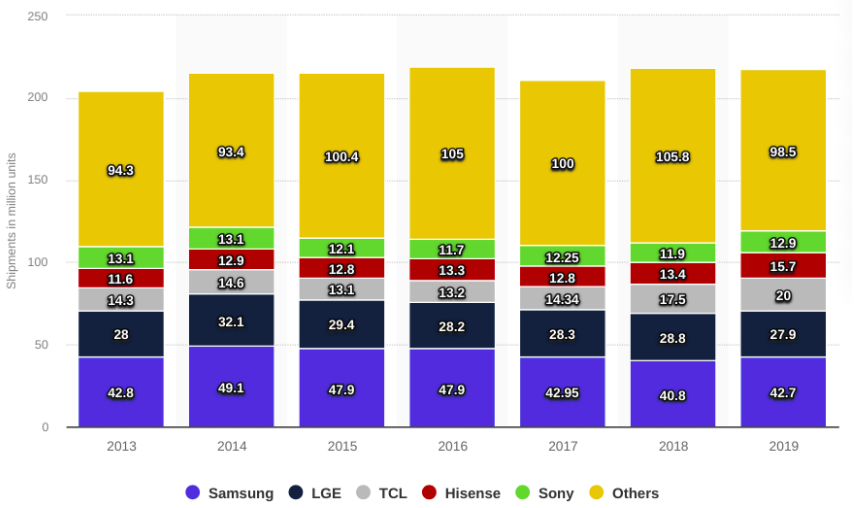

Figure 1.2: Global LCD TV unit shipments from 2015 to 2019, by vendor (in millions), withdrawn from [2]

2 Analysis

In the analysis phase, the product requirements are derived — defining the client expectations for the product — as well as the project constraints — what the environments limits about the product. Finally, the theoretical foundations are outlined, providing the basic technical knowledge to undertake the project.

2.1 Requirements

The requirements defined the client expectations for the TV remote control, namely:

- Remotely operated
- Low weight
- Powered by batteries
- 3 buttons: Power (Off/On); Up and Down for channel selection.
- Infrared emitter response time (system output response time): 100 ms
- The TV remote may be upgraded in the future to use more buttons

2.2 Constraints

The project constraints are the limitations the environment imposes on it, namely:

- the TV remote must contain an infrared emitter (the TV already has an infrared receiver)
- The TV remote control must supply the required data frames imposed by the TV manufacturer
- Data frames may not be provided by the client
- Security concerns are defined by the data frames and the specific communication frequency imposed by the TV manufacturer
- 1 week deadline: 14 h
- Manpower: 2 people
- Budget:
    - HW (parts acquisition and assembly): fixed costs — 1 EUR/unit (1000 batch production)
        - TV remote Shell

  - TV remote membrane
  - Data acquisition & Infrared emitter PCB
- Development: project — 20 EUR per hour per person, totalling 560 EUR + IVA

2.3 Theoretical foundations

The theoretical foundations provide the basic technical knowledge for project undertaking. In that sense, it is important to understand the principle of operation and the related technologies, namely the infrared communication protocol consisting of well-established data frames, specific to each manufacturer and at specific bandwith. It should be highlighted that the communication protocol information is critical for the correct behavior of the TV remote control, as the latter must stimulate the TV, complying to this protocol.

Pushing a button on a remote control sets in motion a series of events that causes the controlled device to carry out a command. The process can be generally described as:

1. pushing the button on the remote control causes a touch to the contact beneath it and complete the button circuit on the circuit board. The integrated circuit detects this.
2. The integrated circuit sends the binary of the button function command to the infrared Light Emitting Diode (LED) at the front of the remote.
3. The LED emits a series of light pulses that corresponds to the binary the button command.

As an example, one can take a look at the clicking on the “volume up” button on a Sony TV remote (Fig. 2.1):

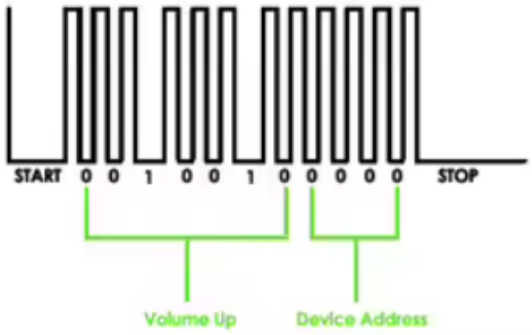

Figure 2.1: Example of wave generator for ”volume up” from [3]

The remote signal includes more than the command for “volume up”. It sends several bits of information to the receiving device, establishing a communication protocol, including:

- a “start” command
- the command code for “volume up”
- the device address (so the TV knows the data is intended for it)

- a “stop” command (triggered when you release the “volume up” button)

In this case, the buttons that are needed and its codes are:

- Power On = 001 0101
- Power Off = 010 1111
- Volume Up = 001 0010
- Volume Down = 001 0011

2.3.1 Reverse engineering

The contract established between the client (Samsung company) and the developer team (the authors) imposes the disclosement of the required information about the communication protocol. However, this is not always necessarily the case. As such, it is important to have a backup plan, which, in this case, corresponds to perform reverse engineering on the communication protocol.

For this endeavour, an “attack” can be performed on the TV, by stimulating it at varying frequencies and set of commands and observing its effect. Obviously, the complexity grows with the number of required commands, but as in this case there are only 3 commands, this can be feasible. Furthermore, this can be bootstrapped by using available TV remote control emulators. An example setup can be connecting an infrared (IR) receiver at a Raspberry Pi, as illustrated in Fig. 2.2 and loading a package, called Linux Infrared Remote Control (LIRC) that allows you to decode and send infrared signals of many (but not all) commonly used remote controls.

The most important part of LIRC is the lircd daemon which decodes IR signals received by the device drivers and provides the information on a socket. It also accepts commands for IR signals to be sent if the hardware supports this [4]. Additionally, the sent IR signals can be used to identify the emitter characteristics, if already present in the database, or simply recorded for later usage. For the present use case, the list of available commands for Samsung TVs can also be obtained for the database for actual TV “attack”.

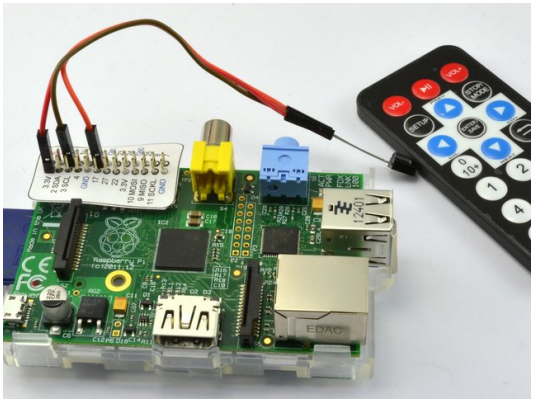

Figure 2.2: Example setup for reverse engineering of TV remote control commands using an emulator [5]

3 Design

In this section the theoretical foundations are used to design a viable solution, accordingly to the requirements and constraints listed. In the design phase, the product development starts, specifying the system in terms of hardware and software and its associated interfaces, the error handling required, and the design verification.

3.1 Hardware specification

The first step for system design is the Hardware (HW) specification. This can be pictured as a block diagram, ideally with Commercial off-the-shelf (COTS) components. Fig. 3.1 depicts the overall information flow and the TV remote block diagram: the **User** interacts with the TV remote by pressing buttons which triggers the emission of IR codes to the TV, containing the IR receiver. As supported by the market research seen in Section 1.2, the block diagram of the TV remote control can be taught of a physical input interface — the pushbuttons, a processing interface — a microprocessor — to determine which keys are pressed and the resulting IR codes to be emitted, and a IR transmitter circuit as the output where the IR LED also works as a visual output. Usually, there will be also a digital interface, to handle the large amount of keys as a Input/Output (I/O) expander with serial interface, e.g. the Microchip MCP23008/MCP23S08 [6], which is considered here for future expansibility of the device, thus making it scalable. This can be especially useful when using microprocessor units with lower I/O inputs. A tradeoff between the inclusion of the multiplexer versus future redesign must be performed. The system is powered by batteries.

The microprocessor chosen depends on various factors, such as: architecture, throughput, memory, processing power, power efficiency, toolchain availability and programming easiness, etc. Additionally, one may consider the usage of a microcontroller, due to added peripherals, easing the Software (SW) burden. For example, the usage of the Programmable Counter Array (PCA) peripheral in the 8051 Micro Controller Unit (MCU) can free the processor from the job of bit-banging the IR transmitter circuitry or the Inter-Integrated Circuit (I2C) or Serial Peripheral Interface (I2C) peripherals to interface the I/O expander. With that said, the choice relied on the 8051 MCU due to its small footprint, clock speed of up to 48 MHz, low power usage on idle, the acquaintance with the toolchain and the programming, and for the inclusion of the PCA, SPI

and I2C peripherals.

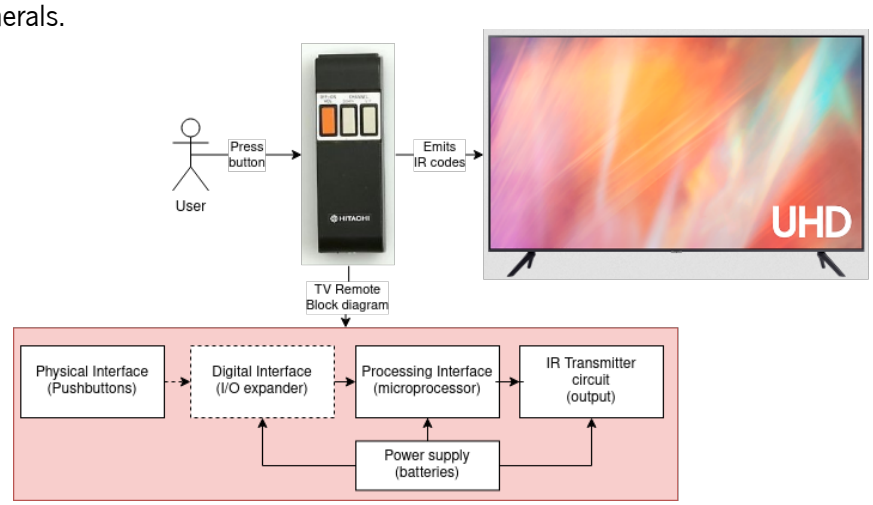

Figure 3.1: Overall information flow and TV remote block diagram

3.2 Hardware interfaces definition

After specifying the HW, it is important to define its interfaces. The TV remote control is clearly an event-driven asynchronous system, thus using HW interrupts. When a user presses a pushbutton that event should be signaled to the Central Processing Unit (CPU) of the MCU via an external interrupt, "waking" it up. The CPU handles that interrupt via an Interrupt Service Routine (ISR), processing it and actuating, if required. The system then goes back to sleep.

The 8051 MCU only contains 2 external interrupts sources — **INT0** and **INT1** — thus limitting the direct connection of the pushbuttons to the MCU, even with the pull-up circuitry. This is where the serial I/O expander becomes most useful, as it can be connected to TV remote keys (up to 8 in this case) and connected to a single external interrupt pin, being then read via I2C or I2C interface. Thus, the keys can be connected through pull-up circuitry to the I/O expander and the output is then connected to **INT0** on the 8051 MCU. The communication protocol chosen is I2C as it is more expandable

Concerning the output, the 8051 PCA peripheral is connected to the IR transmitter circuitry, for IR code emission. The HW interfaces of the system are depicted in Fig. 3.2.

3.3 Software specification

Next, the SW responsible for system operation is specified. As this system is an event-driven asynchronous one it can be more easily specified using a state machine diagram as illustrated in Fig. 3.3. At program startup — **Idle** state — the peripherals are initialized (interrupts, PCA, I2C, timer for tick generated interrupts) and the program waits for events, going into sleep mode.

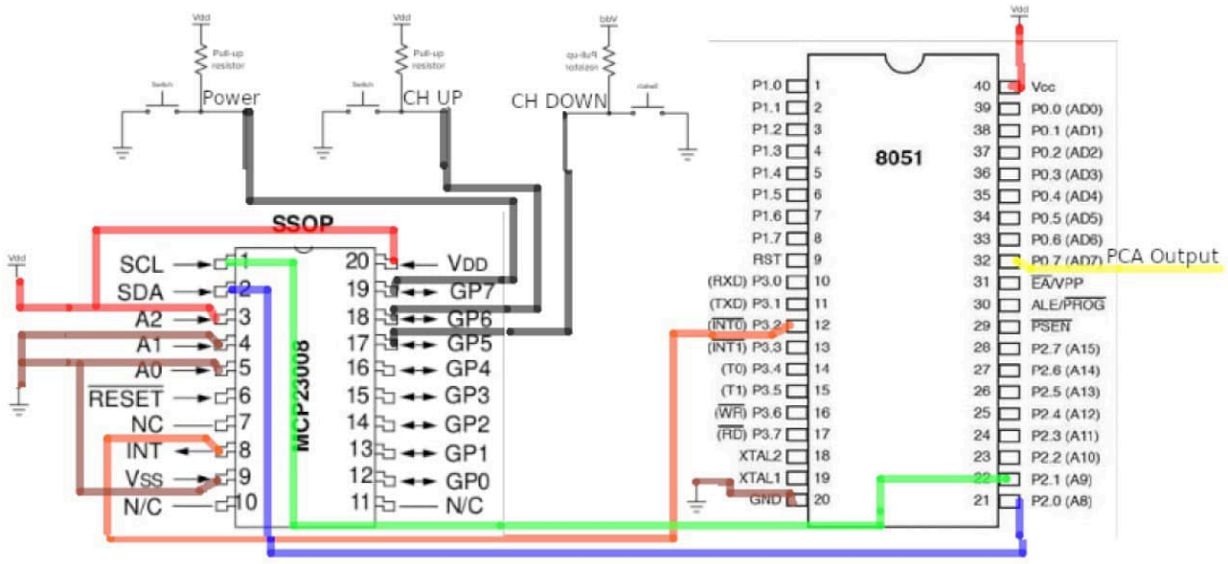

Figure 3.2: HW interfaces of the system

When a key is pressed, the external interrupt **INTO** signals this event right away to the CPU awaking it up triggering the execution of **ISR_Key_Press**. This ISR checks which key was pressed by reading it from the I/O expander via I2C, pushes the associated code to **keycode_fifo** — a circular First-In, First-Out (FIFO) buffer — and flags it to the CPU, enabling **keycode_avail**.

The CPU is awaked regularly to check if any keycode is available. If it is, it triggers the execution of **ISR_IR_emit**, clearing the flag **keycode_avail**, popping the keycode from **keycode_fifo** and sending it via IR circuitry (**ir_send(keycode)**).

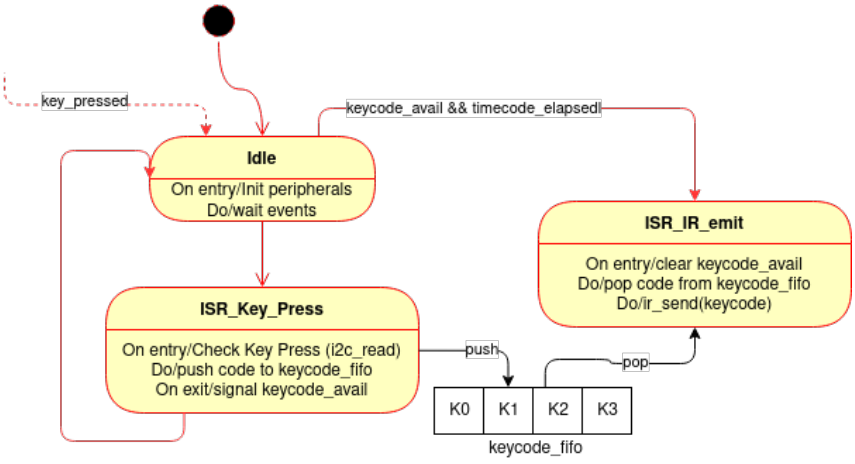

Figure 3.3: TV remote state machine diagram

Fig. 3.4 presents the flowchart for reading from the I/O expander via i2c

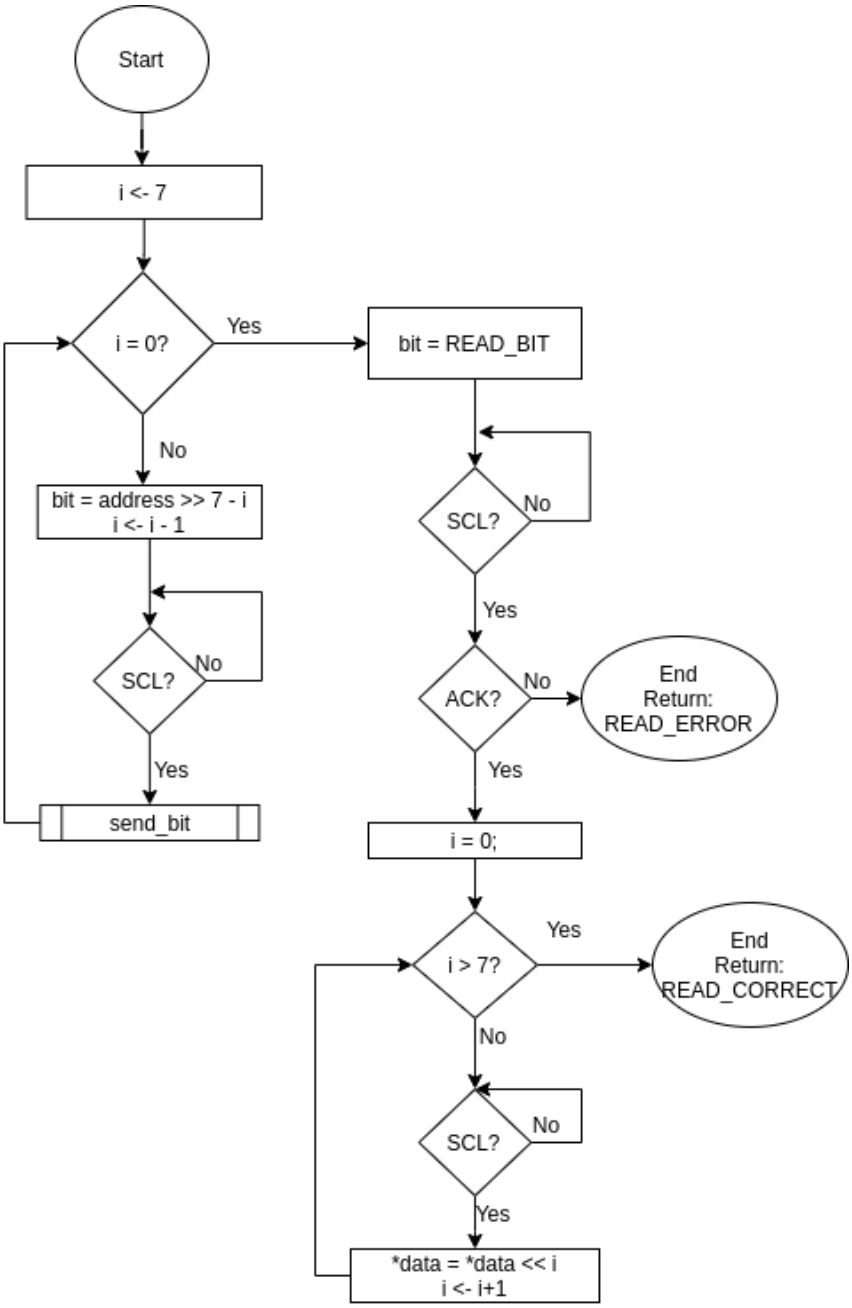

Figure 3.4: I2C Read Flowchart

3.4 Software interfaces definition

From the state machine diagram (Fig. 3.3) the softwares modules and data structures can be infered. The data structure is **keycode_fifo**, a circular buffer to manage the keycodes produced and consumed. The modules are **i2c**, **ir** and **fifo** for determing the key pressed, transmitting the keycode and managing the FIFO

buffer. Enumerations are added for keycodes and errors listing.

An example of the API can be seen in Listing 3.1, using builtin documentation.

```
/**
 * @brief Reads from the designated address a byte
 * @param addr: device address to read from
 * @param data: byte to read
 * @return 0 if success, and the error code otherwise
 */
int i2c_read(char addr, char *data);
/**
 * @brief Push/pop to the buffer a specified command
 * @param fifo: buffer
 * @param data: byte to push/pop/send
 * @return 0 if success, and the error code otherwise
 */
int fifo_push(fifo *f, char data);
int fifo_pop(fifo *f, char *data);
int fifo_init(fifo *fif);
int ir_send(char data);
```

Listing 3.1: API example with builtin documentation

3.5 Start-up/shutdown process specification

As highlighted in Fig. 3.3, the process starts with battery power being supplied to the system, going into sleep mode and waiting for events, minimizing power consumption. However, there is still residual power being drawn. This could be overcome by placing a power button for the remote itself, but with the inconvenience of MCU reset and the initial delays associated. The shutdown results from batteries disconnection.

3.6 Error handling specification

Every system is prone to glitches, bugs and errors, thus, requiring it to be handled. Errors should be handled gracefully by creating error handling routines, which try to circumvent them and provide feedback.

For extreme cases, where this is not possible, the watchdog timer should be enabled to help the system recover from crashes. However, this is a last resort, as constant reboots — sign of a bad design — are inadmissible and will frustrate the user.

The error handling routines can be built into the design by considering return codes and asserts from function calls, e.g., `int i2c_read(char *byte)`, where `0` signals success and otherwise an error was encountered. Thus, good design eases error-handling specification and should be an aspect to keep in mind in this phase.

Bibliography

[1] Infrared remote controls: Inside. URL https://electronics.howstuffworks.com/remote-control1.htm. accessed: 2021-10-11.

[2] Lcd tv shipments worldwide by vendor 2015-2019. URL https://www.statista.com/statistics/668519/lcd-tv-shipments-worldwide-by-vendor/. accessed: 2021-10-11.

[3] Infrared remote controls: The process. URL https://electronics.howstuffworks.com/remote-control2.htm. accessed: 2021-10-11.

[4] Linux infrared remote control. URL https://www.lirc.org/. accessed: 2021-10-11.

[5] Using an ir remote with a raspberry pi media center. URL https://learn.adafruit.com/using-an-ir-remote-with-a-raspberry-pi-media-center. accessed: 2021-10-11.

[6] Mcp23008/mcp23s08 i/o expander with serial interface. URL shorturl.at/jmzV3. accessed: 2021-10-11.