Formation Dual Project 2021
Software Architecture Document

1.Introduction

1. Purpose

   This document describes the architecture of the system through architectural views about Formation Dual Project 2021 with significant architectural decisions.

2. Scope

   This Software Architecture Document provides an architectural overview of the Formation Dual Project 2021. The Formation Dual Project 2021 consists of communication between cards Launchpad C2000 by Texas Instruments, they include the microcontroller TMS320F28377S through SPI protocol, an in pin of mastercard and an out pin of slave.

3. Definitions, Acronyms and Abbreviations

   FDual: Formation Dual

4. References

   - ❏ Material proporcionado por Juan Luis Garcia Camacho
   - ❏ S. Johnson, Software Architecture Document generated using Rational SoDA template and Rational Rose model https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm#Architectural%20Representation

2.Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view and deployment view where it explains what types of model elements it contains.

3.Architectural Goals and Constraints

- The boards shall be Launchpad C2000 by Texas Instruments, they include the microcontroller TMS320F28377S
  The SPI protocol
- Communication speed of 9600 baud
- The data frames will be eight bits.
- The dictionary must include a security message and must determine the actions to be taken
- The parity shall be enabled in the communication
- If the button is pressed once in one second, the output pin will toggle and change its state
- If the button is pressed twice in one second, the output pin shall generate a PWM of 1 Hz
- If the button is pressed three times in one second, the output pin shall generate a PWM of 2 Hz
- The execution of the program must be cyclical in both boards.
- If the button instruction fails, the boards shall reset (system protection)
- If the button is pressed more than 3 times during a second, a red LED shall light up until the system reboots.
- When the program starts the output of the LED must be active and it shall be executing a PWM of 0.5Hz
- The initial 0.5Hz PWM signal shall persist during the executing until another event interrupts it.

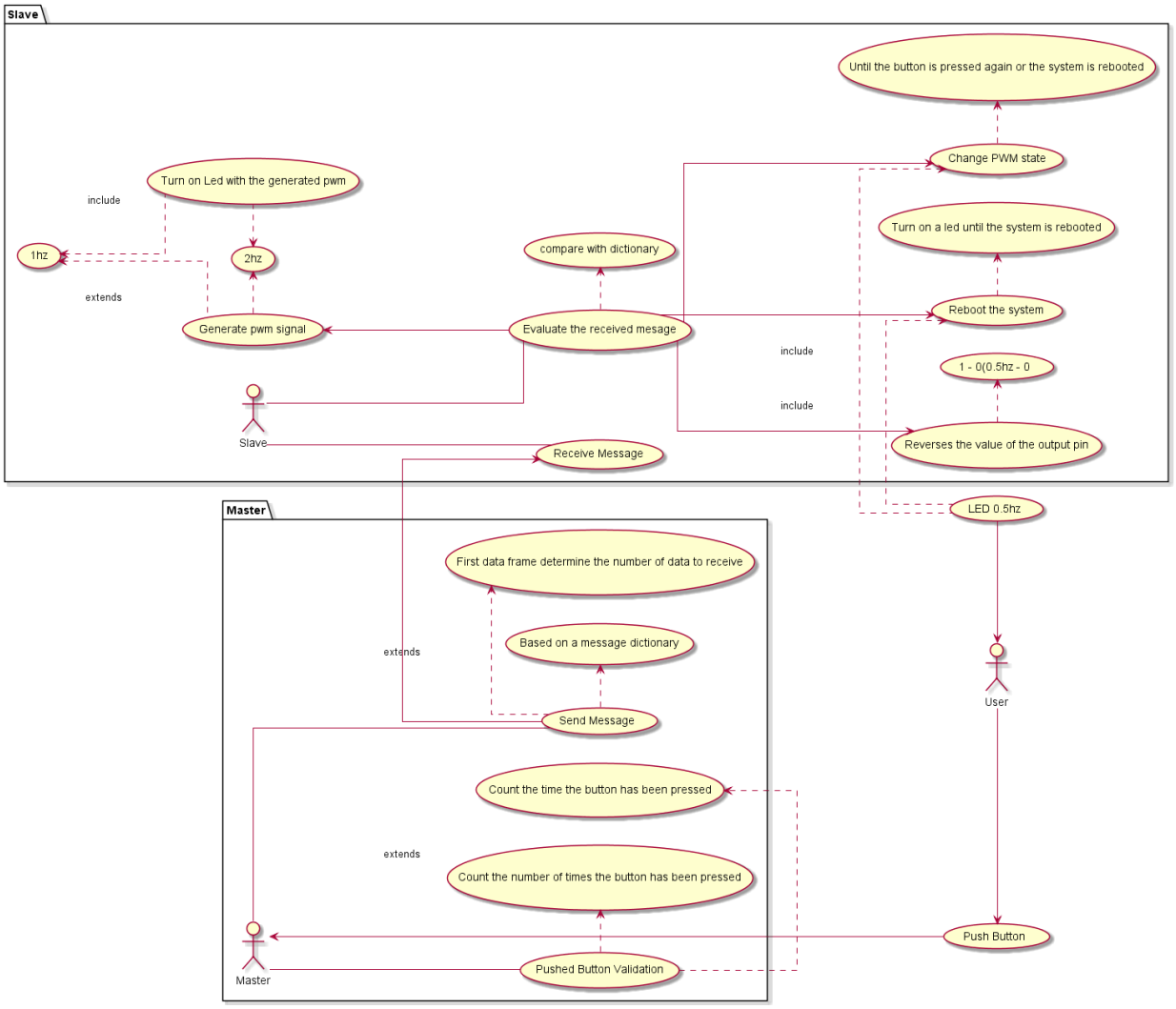
- The PWM shall have a constant 50% duty cycle to module the pulse width generated.

4.Use-Case View

The FDual project cases are:

- Push the button
- Pushed button validation
- Send message
- Receive the message
- Evaluate the received message
- Generate pwm signal
- Reverses the value of the output pin
- Reboot the system
- Change PWM state

4.1.Use-Case Realizations



These cases are initiated by actors like User, Master board and Slave board.

4.1.1.Led ON, 0.5hz.

The system starts with a led in ON state at 0.5hz

4.1.2.Push the button.

When a user pushes the button connected to the master board is going to trigger an instruction. Actor starting this case is user and master

4.1.3.Pushed button validation.

Master board, going to evaluate the instruction entered by the user by pushing the button, will take into account the end time of the pulses and the number of times the button has been pressed.
Actor starting this case is Master board

4.1.4.Send a Message.

When the instruction entered by the button coincides with a validation in the software, the master board will send a message to the slave board according to a dictionary of messages present on both boards.
Actors starting in this case are master and Slave

4.1.5.Receive the message.

The slave board going to receive the message by an specific pin group used for SPI communication
Actors starting in this case are master board and slave board

4.1.5.Evaluate the received message

After receive the message, the slave board going to evaluate the received carácter to execute a specific instruction, the actor in this case is Slave board

4.1.6.Generate PWM signal.

After evaluate the message received the slave board could generate a PWM signal with a duty cycle of 50% and a frequency of 1hz or 2hz, depends of carácter
received by the message, the actor starting in this case is slave board

4.1.7.Reverses the value of the output pin

After evaluating the received message, match with the instruction who reversed the output pin value, the slave board going to set the value of output pin to 0. The actor starting in this case is slave board

4.1.8.Reboot the system

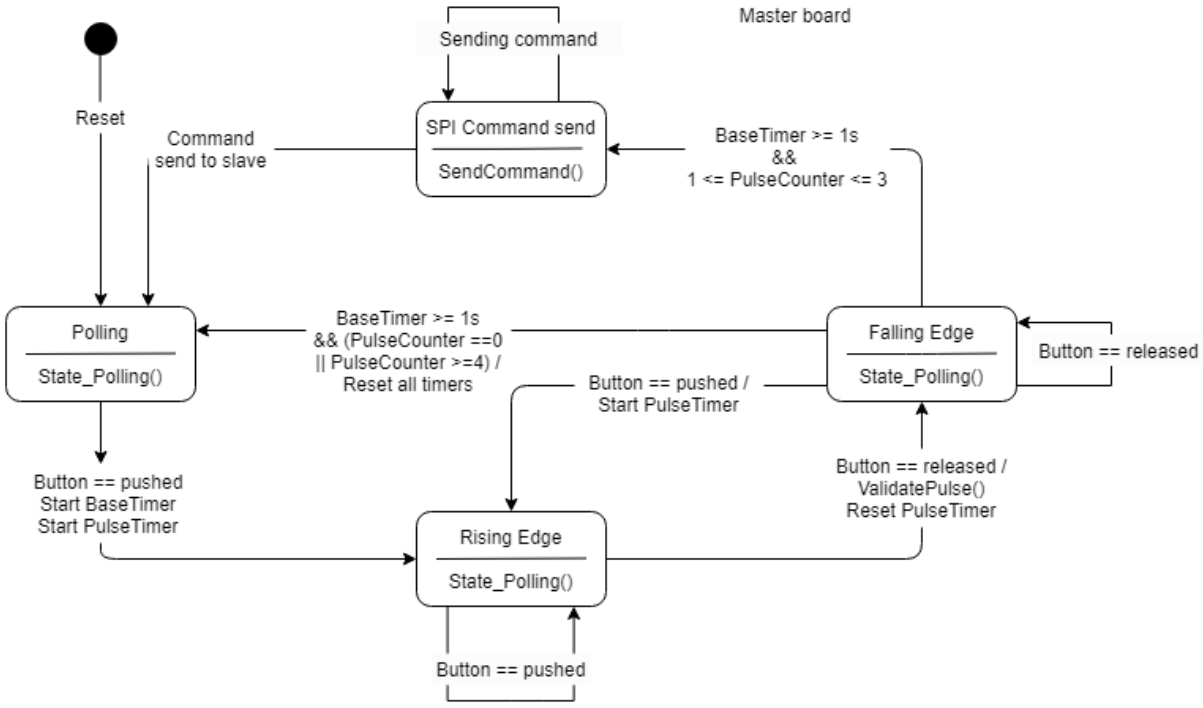
When the received message matches with the carácter who restart the system the slave board is going to Led on until the system is rebooted, the actors in this case are slave board and master board.

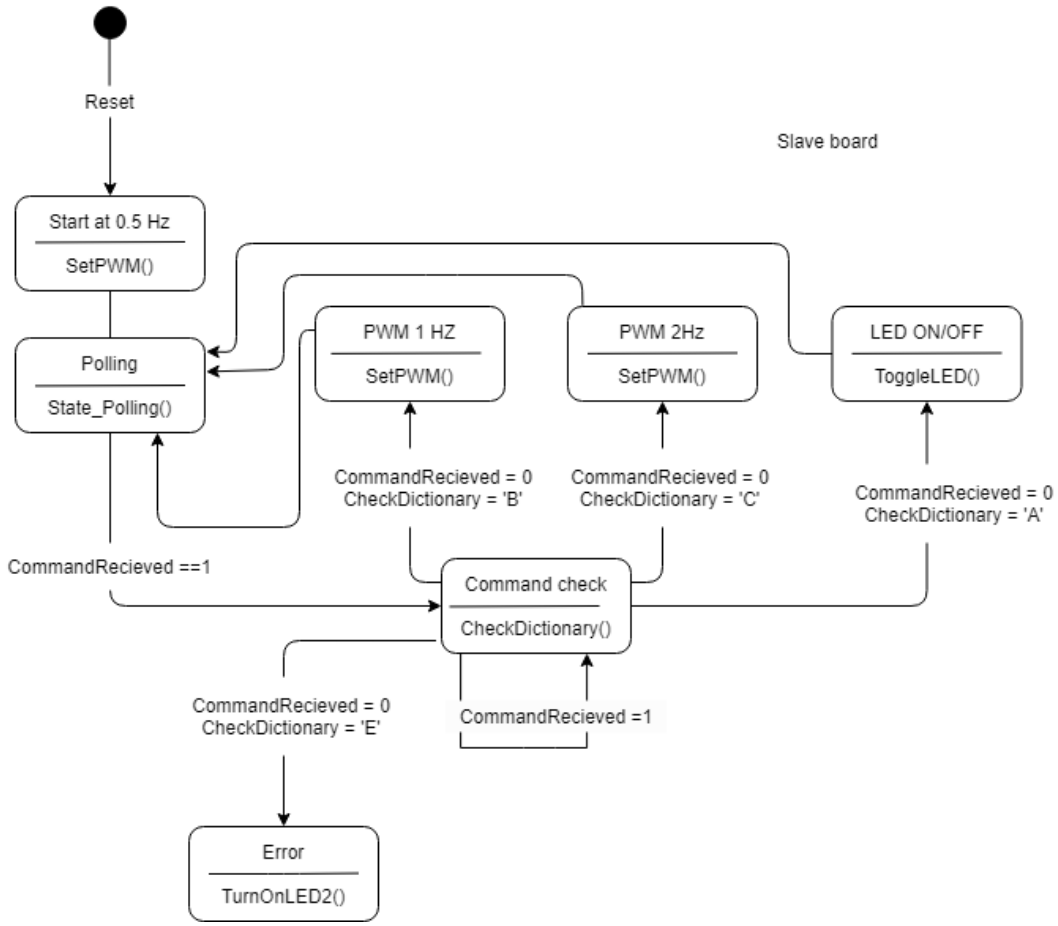
4.1.9.Turn on Led with generate pwm

A Led connected to stay ON after the pwm were generated, the actor in this case is slave board

4.2 State machines

The logic of the boards is shown in the following state machine diagrams.



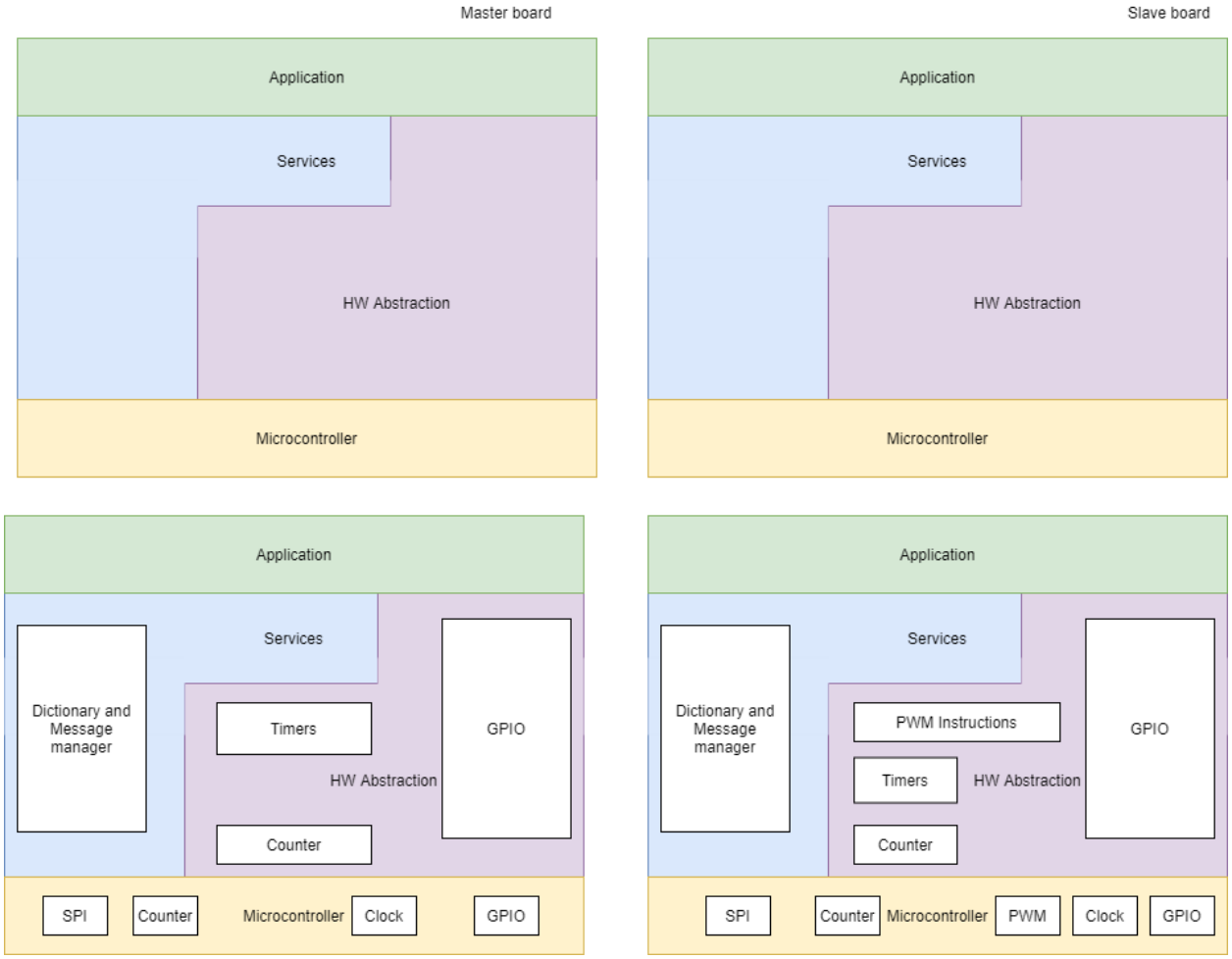


5.Implementation View

This project has been divided in two parts, one for the master board and the second fot the slave one. Both of them share the same general structure in layers.The layers considered in this project consist in Application, Services, HW Abstraction and Microcontroller.

Both boards use SPI communication protocol and they both require the use of timers and counters from the microcontroller but only the slave board uses PWM to module the outputs.

5.1.Overview



The interaction in between layers can only be possible with an immediate layer, in this case, the layer of services will interact directly with all the other ones, but in the HW abstraction one can't share data directly to the application, for example.

- Application layer

  This layer represents the state machine that will be developed by the SW team and that will have the direct interaction with the user. The application layer stores the logic and the algorithm of the project and determines the actions to be executed.

- Services layer

  This layer includes the message manager and will be the key for the communication in between boards and will be used to share the dictionary.

- HW Abstraction layer

  This layer will interpret and manage the registers and the functions of the HW

- Microcontroller

  The microcontroller is the HW in which the algorithm will be stored and executed and need to be managed by the other layers to accomplish the functions needed for the project.

5.2.Layers

5.2.1.Master board

The master board will count and interpret the pulses from the button to send a command via SPI to the slave board and indicate to the slave the action that will be executed

- Application layer

  In the master board this application layer contains the actions that will be part of the coded algorithm

- Services layer

  This layer includes the message manager and will be the key for the communication in between boards and will be used to share the message dictionary.

- HW Abstraction layer

  This layer will interpret and manage the registers and functions of the microcontroller

- Microcontroller

  The microcontroller is the HW in which the algorithm will be stored and need to be managed by the other layers to accomplish the functions needed for the project.

5.2.2.Slave board

- Application layer

  In the master board this application layer contains the actions that will execute the determined commands from the master.

- Services layer

  This layer includes the message manager and will be the key for the communication in between boards and will be used to share this dictionary.

- HW Abstraction layer

  This layer will interpret and manage the registers

- Microcontroller

  The microcontroller is the HW in which the algorithm will be stored and need to be managed by the other layers to accomplish the functions needed for the project.

6.Logical View

The logical view of the project is made up of the 3 main packages: Application layer, Services layer, HW Abstraction layer and Microcontroller.

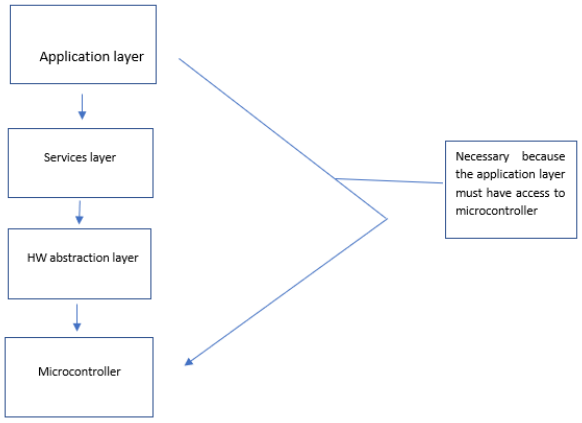
The application layer package contains classes to stores the logic and the algorithm of the project and determines the actions to be executed. The services layer includes classes for sending communication messages using the established dictionary.

The HW Abstraction layer package contains control classes to interact with the system to

interpret the message, control timers, counters and inputs and outputs.

The microcontroller package includes entity classes for the SPI communication, the counters, the clock and the inputs and outputs to the receiver led.

6.1.Overview



- Application layer

  In the master board this application layer contains the actions that will excute the determined commands from the master

- Services layer

  This layer includes the message manager and will be the key for the communication in between boards and will be used to share this dictionary.

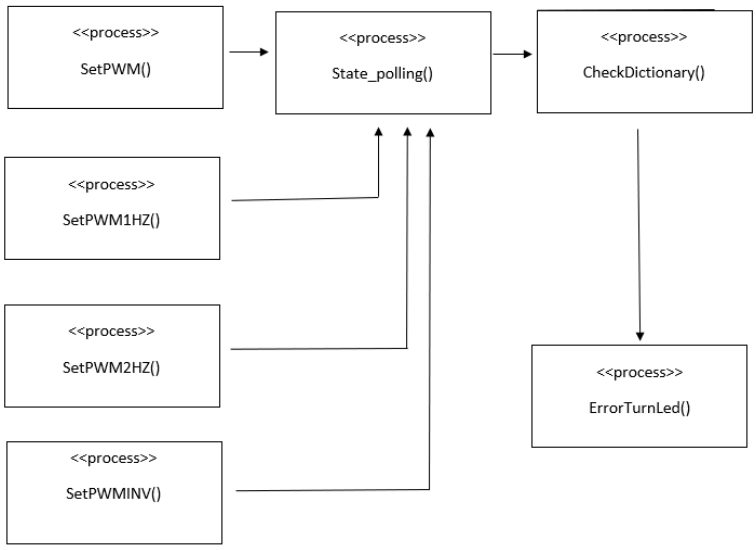
- HW Abstraction layer

  This layer will interpret and manage the registers

- Microcontroller

  The microcontroller is the HW in which the algorithm will be stored and need to be managed by the other layers to accomplish the functions needed for the project.

6.2.Architecturally Significant Design Packages



- SetPWM

Set the pwm to 0.5Hz.

- SetPWM1HZ

Set the pwm to 1Hz.

- SetPWM2HZ

Set the pwm to 2Hz.

- SetPWMINV

Establishes the pwm to the reverse of the current state.

- State_polling

Count the number of pulses.

- CheckDictionary

Compares the data obtained with the established dictionary.

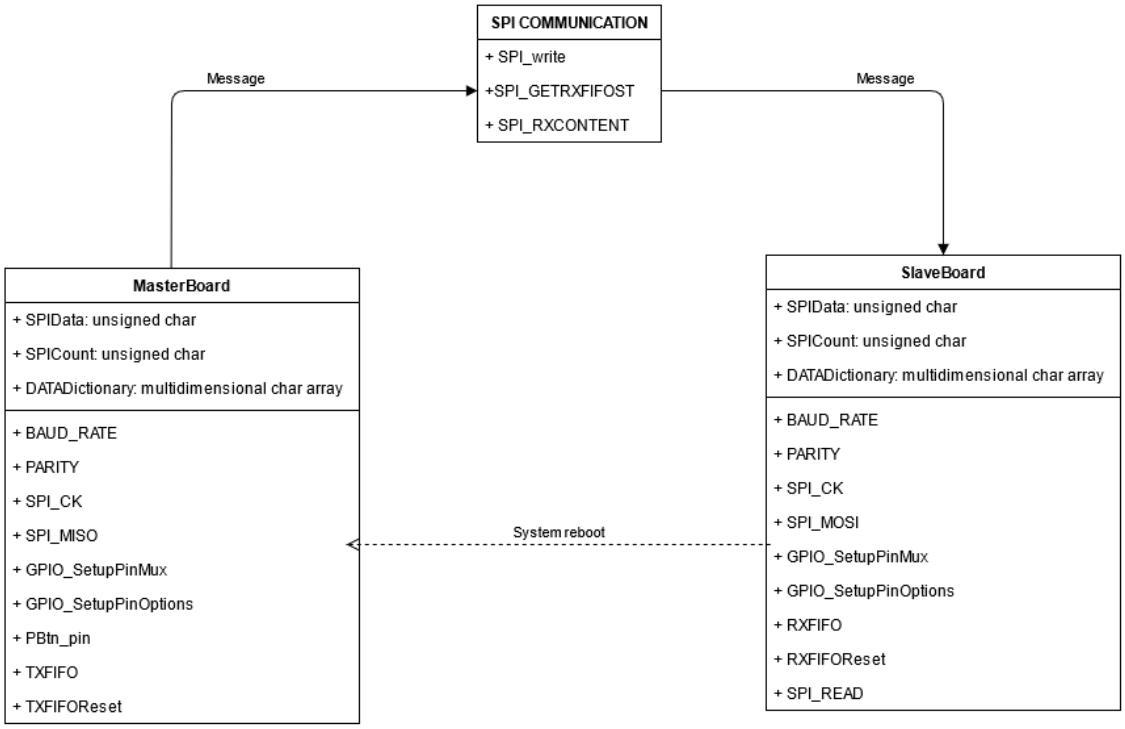
- ErrorTurnLed

Turn on the error led.

7.Process View

This process view illustrate the program code existent in the 2 different board(Master - Slave) by SPI communication protocol, includes the fields and functions declared in each one and the SPI communication
function

7.1



7.1.1 MasterBoard

This process init the communication between the 2 boards that's why you should have instructions like GPIO assignments and Button Pins

FIelds:

- SPIData: UNSIGNED CHAR, Define a data structure for the SPI data
- SPICount: UNSIGNED CHAR, Counter used to clock out the data
- DATADictionary: MULTIDIMENSIONAL CHAR ARRAY, Array used to storage chars to will compare in the slave board

Functions:

- BAUD_RATE, 9600 bits
- PARITY, pair
- SPI_CK, used to set SPI clock

- SPI_MISO, master in slave out
- GPIO_SetupPinMux, GPIO mux select
- GPIO_SetupPinOptions, GPIO pin setting
- PBtn_pin, push button pin input
- TXFIFO, init FIFO
- TXFIFOReset

7.1.2 SPI COMMUNICATION

This process constitutes the sending of the message by spi communication

Functions:

- SPI_write, constitutes the structure of the SPI communication protocol for sending messages
- SPI_GETRXFIFOST, get status of RX
- SPI_RXCONTENT

7.1.3 SlaveBoard

This process read and processes the message sent by master board

FIelds:

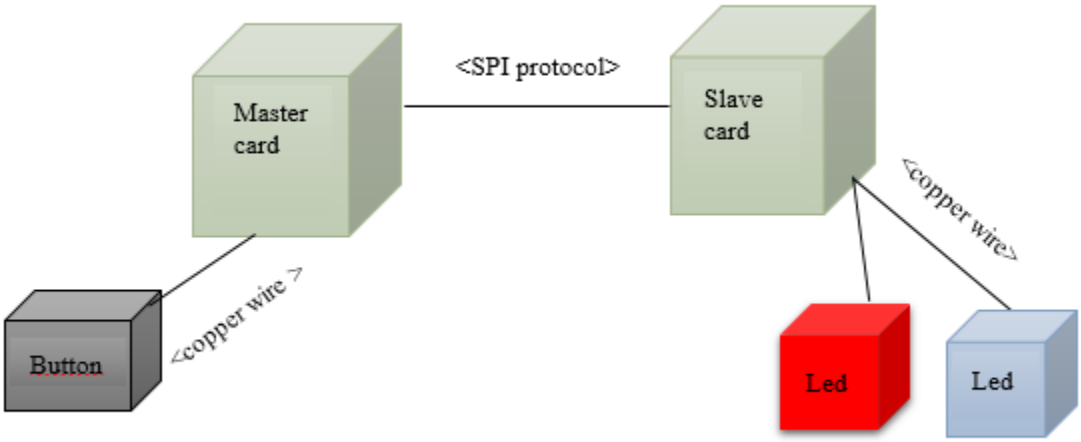
- SPIData: UNSIGNED CHAR, Define a data structure for the SPI data
- SPICount: UNSIGNED CHAR, Counter used to clock out the data
- DATADictionary: MULTIDIMENSIONAL CHAR ARRAY, Array used to storage chars to will compare in the slave board

Functions:

- BAUD_RATE, 9600 bits
- PARITY, pair
- SPI_CK, used to set SPI clock
- SPI_MISO, master in slave out
- GPIO_SetupPinMux, GPIO mux select
- GPIO_SetupPinOptions, GPIO pin setting
- RXFIFO, init FIFO
- RXFIFOReset
- SPI_READ, reads data from master board on the SPI bus

8.Deployment View

The deployment view of the architecture describes the various physical nodes (physical network configuration) for the most typical platform configurations and mapping of tasks to the physical nodes.



8.1 Master card

This card receives a signal of the button and select of the message dictionary what will send to slave card by SPI protocol

8.2 Slave card

This card receives a message by SPI protocol. This message compares with the message dictionary and sends an external signal.

8.3 Button

It's an conditional external signal

8.4 Red Led

It's an error external signal

8.5 Blue Led

It's an external signal indicator

9.Size and Performance

The chosen software architecture shall support sizing and timing as stipulated requirements :

1. Communication speed of 9600 baud
2. If the button is pressed once in a second, the output pin will toggle and change its state
3. If the button is pressed twice in a second, the output pin shall generate a PWM of 1 Hz
4. If the button is pressed three in a second, the outpin pin shall generate a PWM of 2 Hz
5. When the program starts the output of the LED must be active and it shall be executing a PWM of 0.5Hz
6. The initial 0.5Hz PWM signal shall persist during the executing until another event interrupts it.
7. The PWM shall have a constante 50% duty cycle to module the pulse width generated.

The selected architecture supports the sizing and timing requirements through the implementation of a client-server architecture, in this case only timing requirements.

10.Quality

The software architecture supports the quality requirements:

1. If the button is pressed more than 3 times during a second or all time during a second, a red LED shall light up until the system reboots.
2. The execution of the program must be cyclical in both boards
3. The dictionary must include a security message and must determine the actions to be taken