O módulo *Door Controller* é constituído por 3 blocos, sendo dois desses hardware e um dos componentes software. A primeira componente baseia-se num controlador em software, a segunda é um controlador em hardware da porta e a última componente é a porta em questão.

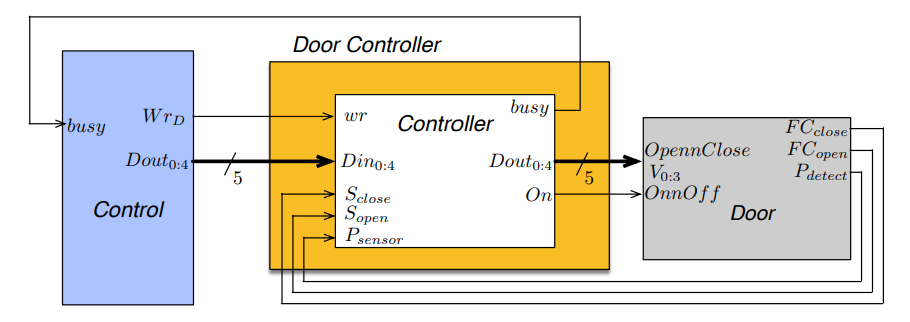
****

Figura 1 – Diagrama de blocos do módulo *Door Controller*

# Controller

O bloco *Controller* implementa um descodificador de um teclado matricial 4x3 por *hardware*, sendo constituído por três sub-blocos: *i)* um teclado matricial de 4x3; *ii)* o bloco *Key Scan,* responsável pelo varrimento do teclado; e *iii)* o bloco *Key Control*, que realiza o controlo do varrimento e o controlo de fluxo, conforme o diagrama de blocos representado na Figura 2a. O controlo de fluxo de saída do bloco *Key Decode* (para o módulo *Key Buffer*), define que o sinal *Kval* é ativado quando é detetada a pressão de uma tecla, sendo também disponibilizado o código dessa tecla no barramento *K0:3*. Apenas é iniciado um novo ciclo de varrimento ao teclado quando o sinal *Kack* for ativado e a tecla premida for libertada. O diagrama temporal do controlo de fluxo está representado na Figura 2b.

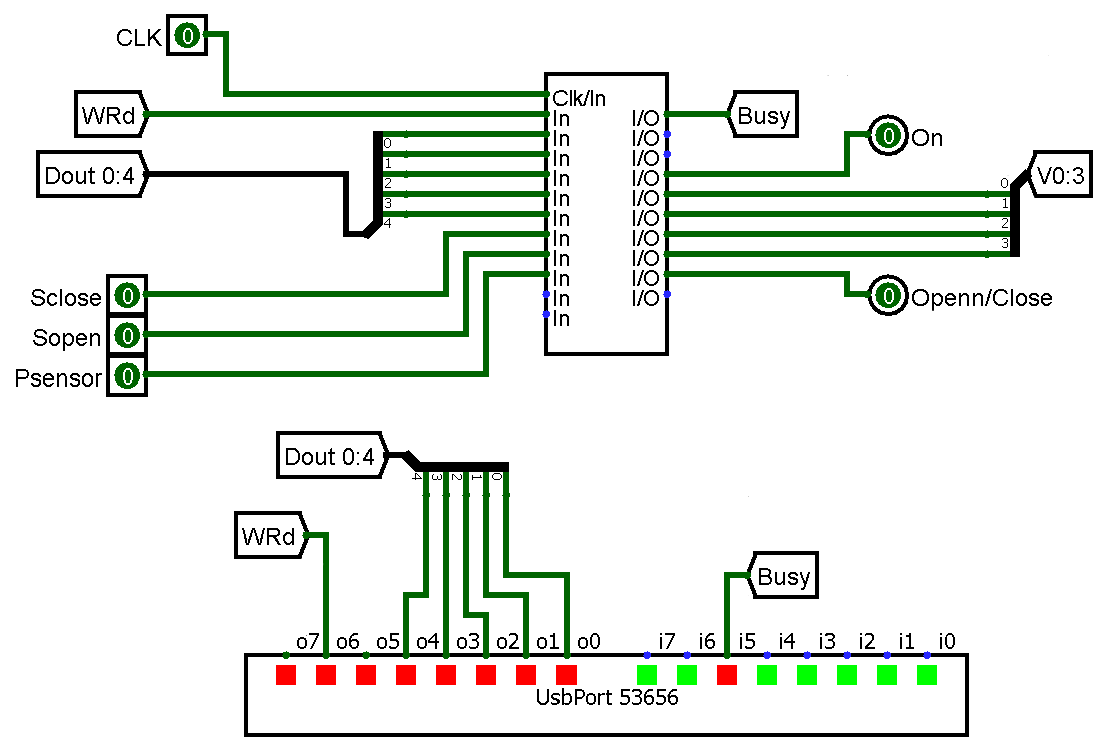


Figura 2 – Bloco *Key Decode*

O bloco *Controller* foi implementado pela máquina de estados representada em *ASM-chart* na 3. **[Adicionar a descrição da solução apresentada.]**

A descrição hardware do bloco *Controller* em CUPL encontra-se no Anexo A.

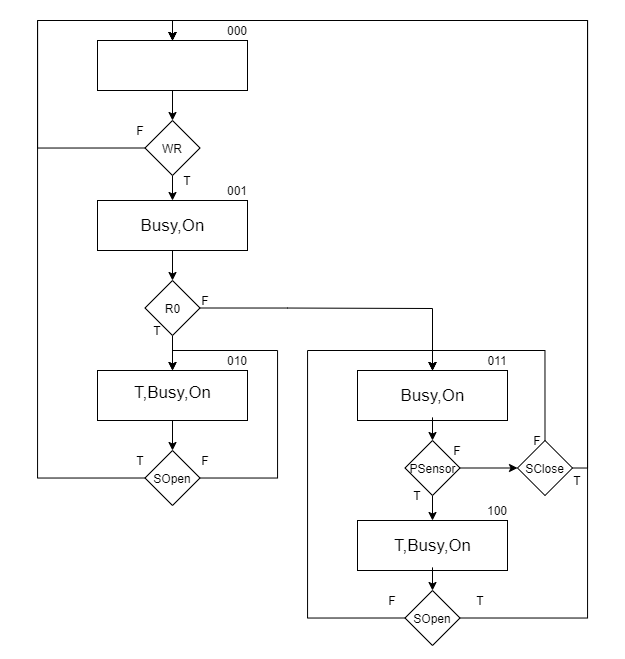


Figura 3– Máquina de estados do bloco *Controller*

Com base nas descrições do bloco *Controller*  implementou-se parcialmente o módulo *Door Controller* de acordo com o esquema elétrico representado no Anexo C. **[Justificar as opções tomadas, como por exemplo, o valor das resistências, as frequências de relógio, etc.]**

# Interface com o *Control*

Implementou-se o módulo *Control* em *software*, recorrendo a linguagem *Kotlin* e seguindo a arquitetura lógica apresentada na Figura 4.





Figura 4 – Diagrama lógico do módulo *Control* de interface com o módulo *Door Controller*

A classe *Door* desenvolvidas é descrita na secções 2.1. , e o código fonte desenvolvido no Anexo C.

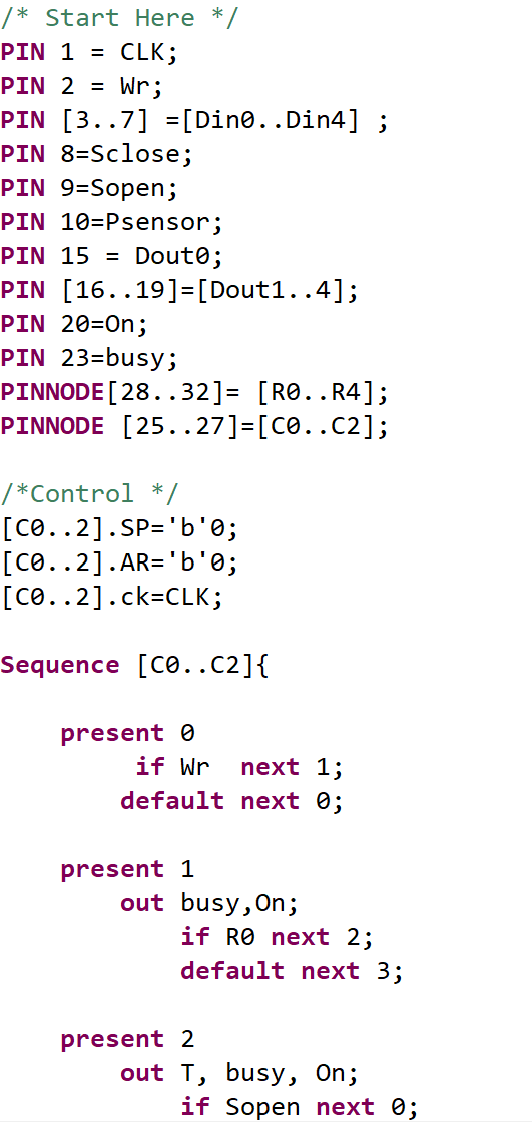
## Classe *Door*

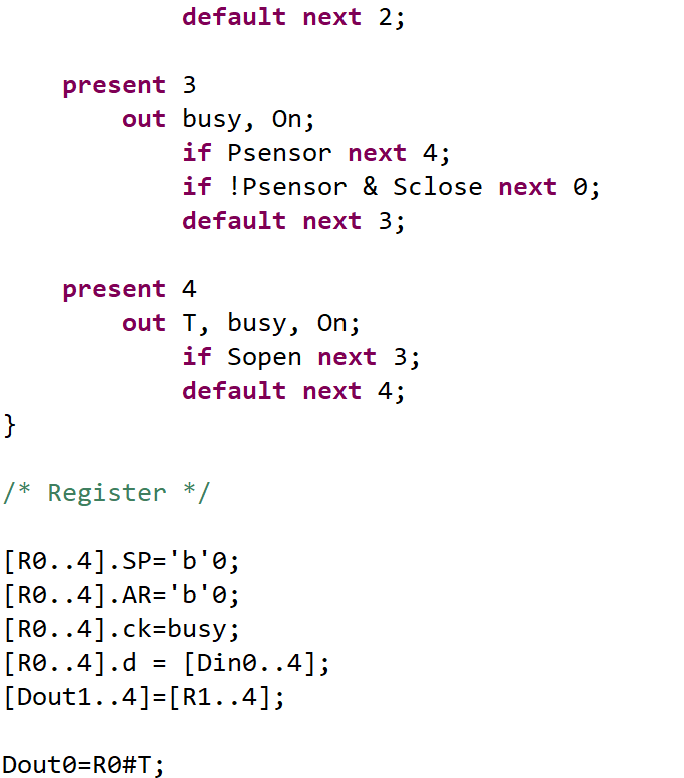
**[Descrever nesta secção de forma sucinta a classe, referindo se adicionaram métodos.]**

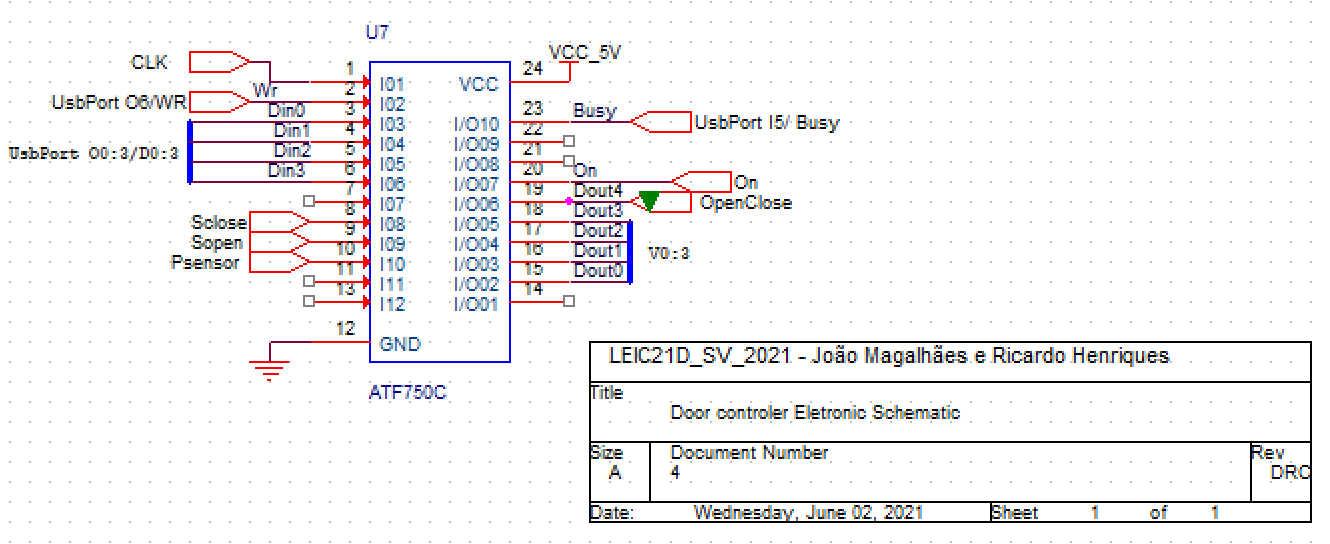
# Conclusões

**[Escrever nesta secção as conclusões da implementação do modulo, incluindo os recursos utilizados, latência na de deteção de tecla, etc.]**

2. Descrição CUPL do bloco *Controller*





1. Esquema elétrico do módulo *Door Controller*

1. Código *Kotlin* da classe *Door*

object Door{

private const val WrMask = 0x40

private const val BusyMask = 0x20

private const val DOutMask = 0x1F

private const val MaxSpeed = 0x0F

fun init(){

HAL.clrBits(WrMask)

}

/\*\*

\* V0~3 -> 0x0F

\*

\* OpenClose -> OPEN = 0x10 & CLOSE = 0x00

\*\*/

fun open(speed:Int){

var spd=speed

if (spd > MaxSpeed) spd=MaxSpeed

val x = 0x10 + spd /\*Open action + speed\*/

HAL.writeBits(DOutMask,x)

HAL.setBits(WrMask)

while (!HAL.isBit(BusyMask)){} /\*Waiting for the busy signal\*/

HAL.clrBits(WrMask)

}

fun close(speed: Int){

var spd=speed

if (spd > MaxSpeed) spd=MaxSpeed

val x = 0x00 + spd /\*Close action + speed\*/

HAL.writeBits(DOutMask,x)

HAL.setBits(WrMask)

while (!HAL.isBit(BusyMask)){} /\*Waiting for the busy signal\*/

HAL.clrBits(WrMask)

}

fun isFinished():Boolean{

while (HAL.isBit(BusyMask)) {

return false

}

return true

}

}

fun main(){

Door.init()

while (true) {

val x = (-100..100).random() /\* TestCode to give random number so door open or close with the value of x\*/

println(x)

if (x <= 0) {

Door.close(12)

} else {

Door.open(12)

}

while (!Door.isFinished()){}

}

}