

Sistema de seguridad en cámaras frigoríficas

Generado por Doxygen 1.12.0

1 Índice jerárquico	1
1.1 Jerarquía de clases	1
2 Índice de estructuras de datos	3
2.1 Estructuras de datos	3
3 Índice de archivos	5
3.1 Lista de archivos	5
4 Documentación de estructuras de datos	7
4.1 Referencia de la clase accesoRFID	7
4.1.1 Descripción detallada	8
4.1.2 Documentación de constructores y destructores	8
4.1.2.1 accesoRFID()	8
4.2 Referencia de la clase DHT11	8
4.2.1 Descripción detallada	9
4.2.2 Documentación de constructores y destructores	9
4.2.2.1 DHT11()	9
4.2.3 Documentación de funciones miembro	9
4.2.3.1 readHumidity()	9
4.2.3.2 readTemperature()	10
4.2.3.3 readTemperatureHumidity()	10
4.2.3.4 setDelay()	10
4.2.4 Documentación de símbolos amigos y relacionados	10
4.2.4.1 setPinDHT11	10
4.3 Referencia de la clase MFRC522	11
4.3.1 Documentación de estructuras de datos	15
4.3.1.1 struct MFRC522::MIFARE_Key	15
4.3.1.2 struct MFRC522::Uid	16
4.4 Referencia de la clase MFRC522Extended	16
4.4.1 Descripción detallada	22
4.4.2 Documentación de estructuras de datos	22
4.4.2.1 struct MFRC522Extended::Ats	22
4.4.2.2 struct MFRC522Extended::Ats.ta1	23
4.4.2.3 struct MFRC522Extended::Ats.tb1	24
4.4.2.4 struct MFRC522Extended::Ats.tc1	24
4.4.2.5 struct MFRC522Extended::PcbBlock	25
4.4.2.6 struct MFRC522Extended::PcbBlock.inf	25
4.4.2.7 struct MFRC522Extended::PcbBlock.prologue	26
4.4.2.8 struct MFRC522Extended::TagInfo	27
4.4.3 Documentación de funciones miembro	27
4.4.3.1 PICC_IsNewCardPresent()	27
4.4.3.2 PICC_ReadCardSerial()	27

4.4.3.3 PICC_Select()	28
4.5 Referencia de la clase timer	28
4.5.1 Descripción detallada	28
4.5.2 Documentación de constructores y destructores	29
4.5.2.1 timer()	29
5 Documentación de archivos	31
5.1 DHT11.h	31
5.2 Referencia del archivo main/libraries/DHT11-HW-481/setPin.cpp	31
5.2.1 Descripción detallada	32
5.2.2 Documentación de funciones	32
5.2.2.1 setPinDHT11()	32
5.3 Referencia del archivo main/libraries/DHT11-HW-481/setPin.h	33
5.3.1 Descripción detallada	33
5.3.2 Documentación de funciones	33
5.3.2.1 setPinDHT11()	33
5.4 setPin.h	34
5.5 accesorRFID.h	34
5.6 deprecated.h	34
5.7 MFRC522.h	34
5.8 MFRC522Extended.h	39
5.9 require_cpp11.h	40
5.10 Referencia del archivo main/libraries/TIMER/timer.h	40
5.10.1 Descripción detallada	41
5.11 timer.h	41
Índice alfabético	43

Capítulo 1

Índice jerárquico

1.1. Jerarquía de clases

Este listado de herencia está ordenado de forma general pero no está en orden alfabético estricto:

accesoRFID	7
MFRC522Extended::Ats	16
MFRC522Extended::Ats.ta1	16
MFRC522Extended::Ats.tb1	16
MFRC522Extended::Ats.tc1	16
DHT11	8
MFRC522	11
MFRC522Extended	16
MFRC522::MIFARE_Key	11
MFRC522Extended::PcbBlock	16
MFRC522Extended::PcbBlock.inf	16
MFRC522Extended::PcbBlock.prologue	16
MFRC522Extended::TagInfo	16
timer	28
MFRC522::Uid	11

Capítulo 2

Índice de estructuras de datos

2.1. Estructuras de datos

Lista de estructuras con breves descripciones:

accesoRFID	
Clase para manejar el acceso usando el módulo RFID-RC522	7
DHT11	8
MFRC522	11
MFRC522Extended	16
timer	
Clase para gestionar un temporizador con dos salidas, cada una es activada en intervalos distintos, con la funcionalidad de antirrebote en los pines de ingreso y reset	28

Capítulo 3

Índice de archivos

3.1. Lista de archivos

Lista de todos los archivos documentados y con breves descripciones:

main/libraries/DHT11-HW-481/ DHT11.h	31
main/libraries/DHT11-HW-481/ setPin.cpp	
Función amiga setPinDHT11	31
main/libraries/DHT11-HW-481/ setPin.h	33
main/libraries/RFID-RC522/ accesoRFID.h	34
main/libraries/RFID-RC522/ deprecated.h	34
main/libraries/RFID-RC522/ MFRC522.h	34
main/libraries/RFID-RC522/ MFRC522Extended.h	39
main/libraries/RFID-RC522/ require_cpp11.h	40
main/libraries/TIMER/ timer.h	
Declaración de la clase timer para manejar dos fases de un temporizador, con antirrebote y un reset	40

Capítulo 4

Documentación de estructuras de datos

4.1. Referencia de la clase accesoRFID

Clase para manejar el acceso usando el módulo RFID-RC522.

```
#include <accesoRFID.h>
```

Diagrama de colaboración de accesoRFID:



Métodos públicos

- `accesoRFID` (byte ssPin, byte rstPin, String autorizarID)
Constructor de la clase `accesoRFID`.
- void `start` ()
void `start()`; Inicia la comunicación SPI con el módulo RFID (usando sus métodos).
- void `mostrarID` ()
void `mostrarID()`; Imprime por puerto serie el UID de la tarjeta detectada por el RFID
- bool `autorizar` ()
bool `autorizar()`; Compara el UID de la tarjeta con el UID autorizado. Devuelve true (acceso permitido) o un false (acceso denegado).
- bool `detectarTarjeta` ()
bool `detectarTarjeta()`; Lee el UID de la tarjeta para implementarla en el sistema.

4.1.1. Descripción detallada

Clase para manejar el acceso usando el módulo RFID-RC522.

Esta clase permite inicializar el lector RFID y verificar si una tarjeta tiene autorización para entrar.

4.1.2. Documentación de constructores y destructores

4.1.2.1. accesoRFID()

```
accesoRFID::accesoRFID (
    byte ssPin,
    byte rstPin,
    String autorizarID)
```

Constructor de la clase [accesoRFID](#).

Parámetros

<i>ssPin</i>	El pin SS (Slave Select) para la comunicación SPI.
<i>rstPin</i>	es el pin de Reset para el módulo RFID.
<i>autorizarID</i>	El UID de la tarjeta autorizada para el acceso.

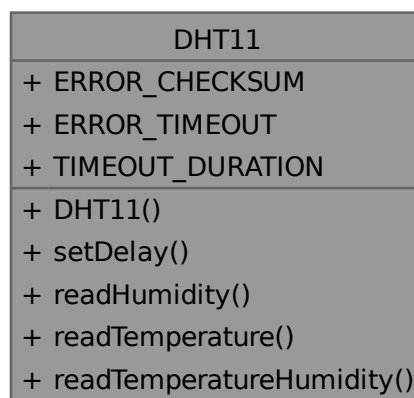
La documentación de esta clase está generada del siguiente archivo:

- main/libraries/RFID-RC522/accesoRFID.h

4.2. Referencia de la clase DHT11

```
#include <DHT11.h>
```

Diagrama de colaboración de DHT11:



Métodos públicos

- `DHT11` (int pin)
- void `setDelay` (unsigned long delay)
- int `readHumidity` ()
- int `readTemperature` ()
- int `readTemperatureHumidity` (int &temperature, int &humidity)

Atributos públicos estáticos

- static const int `ERROR_CHECKSUM` = 254
- static const int `ERROR_TIMEOUT` = 253
- static const int `TIMEOUT_DURATION` = 1000

Amigas

- void `setPinDHT11` (`DHT11` &sensor, int pin)

4.2.1. Descripción detallada

`DHT11.h` Header file for the `DHT11` library, providing functionalities to interface with the `DHT11` temperature & humidity sensor.

Author: Dhruva Saha Version: 2.1.0 License: MIT `DHT11` Class Provides methods to read temperature and humidity data from the `DHT11` sensor.

4.2.2. Documentación de constructores y destructores

4.2.2.1. `DHT11()`

```
DHT11::DHT11 (
    int pin)
```

Constructor Initializes the data pin to be used for communication with the `DHT11` sensor.

Parámetros

<code>pin</code>	Digital pin number on the Arduino board to which the <code>DHT11</code> sensor is connected.
------------------	--

4.2.3. Documentación de funciones miembro

4.2.3.1. `readHumidity()`

```
int DHT11::readHumidity ()
```

Reads and returns the humidity from the `DHT11` sensor.

Devuelve

: Humidity value in percentage. Returns `DHT11_ERROR_TIMEOUT` if reading times out. Returns `DHT11_ERROR_CHECKSUM` if checksum validation fails.

4.2.3.2. readTemperature()

```
int DHT11::readTemperature ()
```

Reads and returns the temperature from the [DHT11](#) sensor.

Devuelve

: Temperature value in Celsius. Returns DHT11_ERROR_TIMEOUT if reading times out. Returns DHT11_ERROR_CHECKSUM if checksum validation fails.

4.2.3.3. readTemperatureHumidity()

```
int DHT11::readTemperatureHumidity (
    int & temperature,
    int & humidity)
```

Reads and returns the temperature and humidity from the [DHT11](#) sensor.

Parámetros

<i>temperature</i>	Reference to a variable where the temperature value will be stored.
<i>humidity</i>	Reference to a variable where the humidity value will be stored.

Devuelve

: true if the reading is successful, false if it fails due to timeout or checksum error.

4.2.3.4. setDelay()

```
void DHT11::setDelay (
    unsigned long delay)
```

Sets the delay between consecutive sensor readings. If this method is not called, a default delay of 500 milliseconds is used.

Parámetros

<i>delay</i>	Delay duration in milliseconds between sensor readings.
--------------	---

4.2.4. Documentación de símbolos amigos y relacionados

4.2.4.1. setPinDHT11

```
void setPinDHT11 (
    DHT11 & sensor,
    int pin) [friend]
```

Bloque if-else Controla si el pin ingresado es uno físicamente válido. Imprime por puerto serie UART un mensaje de error en caso de no serlo.

pinMode(sensor._pin, INPUT) "Desactiva" el pin anterior.

sensor._pin = pin Actualiza el pin.

pinMode(pin, OUTPUT) y digitalWrite(pin, HIGH) Inicializa el nuevo pin como lo hace internamente la clase [DHT11](#)

delay(250) Tiempo de espera a que el sensor se estabilice.

La documentación de esta clase está generada del siguiente archivo:

- main/libraries/DHT11-HW-481/DHT11.h

4.3. Referencia de la clase MFRC522

Diagrama de herencia de MFRC522

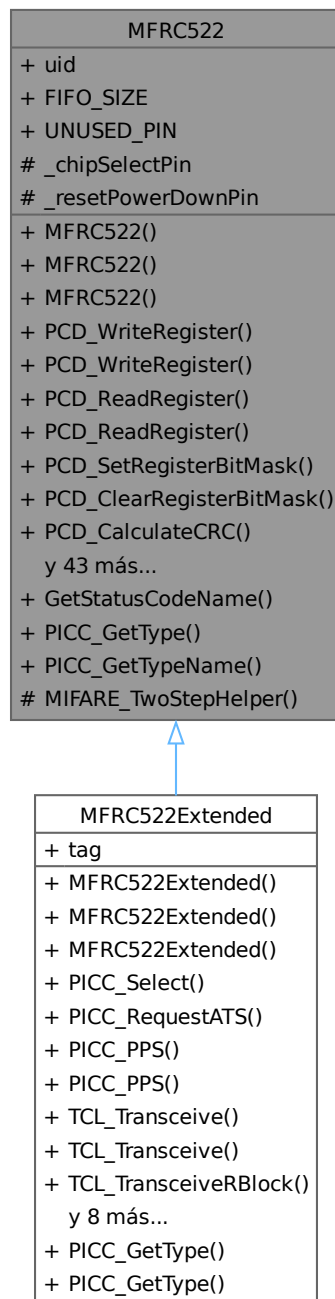


Diagrama de colaboración de MFRC522:



Estructuras de datos

- struct [MIFARE_Key](#)
- struct [Uid](#)

Tipos públicos

- enum **PCD_Register** : byte {
CommandReg = 0x01 << 1 , **ComIrEnReg** = 0x02 << 1 , **DivIrEnReg** = 0x03 << 1 , **ComIrIrqReg** = 0x04


```

<< 1 ,
DivIrqReg = 0x05 << 1 , ErrorReg = 0x06 << 1 , Status1Reg = 0x07 << 1 , Status2Reg = 0x08 << 1 ,
FIFODataReg = 0x09 << 1 , FIFOLevelReg = 0x0A << 1 , WaterLevelReg = 0x0B << 1 , ControlReg =
0x0C << 1 ,
BitFramingReg = 0x0D << 1 , CollReg = 0x0E << 1 , ModeReg = 0x11 << 1 , TxModeReg = 0x12 <<
1 ,
RxModeReg = 0x13 << 1 , TxControlReg = 0x14 << 1 , TxASKReg = 0x15 << 1 , TxSelReg = 0x16 <<
1 ,
RxSelReg = 0x17 << 1 , RxThresholdReg = 0x18 << 1 , DemodReg = 0x19 << 1 , MfTxReg = 0x1C
<< 1 ,
MfRxReg = 0x1D << 1 , SerialSpeedReg = 0x1F << 1 , CRCResultRegH = 0x21 << 1 , CRCResultRegL
= 0x22 << 1 ,
ModWidthReg = 0x24 << 1 , RFCfgReg = 0x26 << 1 , GsNReg = 0x27 << 1 , CWGsPReg = 0x28 <<
1 ,
ModGsPReg = 0x29 << 1 , TModeReg = 0x2A << 1 , TPrescalerReg = 0x2B << 1 , TReloadRegH =
0x2C << 1 ,
TReloadRegL = 0x2D << 1 , TCounterValueRegH = 0x2E << 1 , TCounterValueRegL = 0x2F << 1 ,
TestSel1Reg = 0x31 << 1 ,
TestSel2Reg = 0x32 << 1 , TestPinEnReg = 0x33 << 1 , TestPinValueReg = 0x34 << 1 , TestBusReg
= 0x35 << 1 ,
AutoTestReg = 0x36 << 1 , VersionReg = 0x37 << 1 , AnalogTestReg = 0x38 << 1 , TestDAC1Reg =
0x39 << 1 ,
TestDAC2Reg = 0x3A << 1 , TestADCReg = 0x3B << 1 }
■ enum PCD_Command : byte {
PCD_Idle = 0x00 , PCD_Mem = 0x01 , PCD_GenerateRandomID = 0x02 , PCD_CalcCRC = 0x03 ,
PCD_Transmit = 0x04 , PCD_NoCmdChange = 0x07 , PCD_Receive = 0x08 , PCD_Transceive = 0x0C ,
PCD_MFAuthent = 0x0E , PCD_SoftReset = 0x0F }
■ enum PCD_RxGain : byte {
RxGain_18dB = 0x00 << 4 , RxGain_23dB = 0x01 << 4 , RxGain_18dB_2 = 0x02 << 4 , RxGain_↵
23dB_2 = 0x03 << 4 ,
RxGain_33dB = 0x04 << 4 , RxGain_38dB = 0x05 << 4 , RxGain_43dB = 0x06 << 4 , RxGain_48dB =
0x07 << 4 ,
RxGain_min = 0x00 << 4 , RxGain_avg = 0x04 << 4 , RxGain_max = 0x07 << 4 }
■ enum PICC_Command : byte {
PICC_CMD_REQA = 0x26 , PICC_CMD_WUPA = 0x52 , PICC_CMD_CT = 0x88 , PICC_CMD_SEL_CL1 =
0x93 ,
PICC_CMD_SEL_CL2 = 0x95 , PICC_CMD_SEL_CL3 = 0x97 , PICC_CMD_HLTA = 0x50 , PICC_CMD_↵
RATS = 0xE0 ,
PICC_CMD_MF_AUTH_KEY_A = 0x60 , PICC_CMD_MF_AUTH_KEY_B = 0x61 , PICC_CMD_MF_READ
= 0x30 , PICC_CMD_MF_WRITE = 0xA0 ,
PICC_CMD_MF_DECREMENT = 0xC0 , PICC_CMD_MF_INCREMENT = 0xC1 , PICC_CMD_MF_↵
RESTORE = 0xC2 , PICC_CMD_MF_TRANSFER = 0xB0 ,
PICC_CMD_UL_WRITE = 0xA2 }
■ enum MIFARE_Misc { MF_ACK = 0xA , MF_KEY_SIZE = 6 }
■ enum PICC_Type : byte {
PICC_TYPE_UNKNOWN , PICC_TYPE_ISO_14443_4 , PICC_TYPE_ISO_18092 , PICC_TYPE_↵
MIFARE_MINI ,
PICC_TYPE_MIFARE_1K , PICC_TYPE_MIFARE_4K , PICC_TYPE_MIFARE_UL , PICC_TYPE_↵
MIFARE_PLUS ,
PICC_TYPE_MIFARE_DESFIRE , PICC_TYPE_TNP3XXX , PICC_TYPE_NOT_COMPLETE = 0xff }
■ enum StatusCode : byte {
STATUS_OK , STATUS_ERROR , STATUS_COLLISION , STATUS_TIMEOUT ,
STATUS_NO_ROOM , STATUS_INTERNAL_ERROR , STATUS_INVALID , STATUS_CRC_WRONG ,
STATUS_MIFARE_NACK = 0xff }

```

Métodos públicos

- MFRC522 (byte resetPowerDownPin)

- **MFRC522** (byte chipSelectPin, byte resetPowerDownPin)
- void **PCD_WriteRegister** (PCD_Register reg, byte value)
- void **PCD_WriteRegister** (PCD_Register reg, byte count, byte *values)
- byte **PCD_ReadRegister** (PCD_Register reg)
- void **PCD_ReadRegister** (PCD_Register reg, byte count, byte *values, byte rxAlign=0)
- void **PCD_SetRegisterBitMask** (PCD_Register reg, byte mask)
- void **PCD_ClearRegisterBitMask** (PCD_Register reg, byte mask)
- StatusCode **PCD_CalculateCRC** (byte *data, byte length, byte *result)
- void **PCD_Init** ()
- void **PCD_Init** (byte resetPowerDownPin)
- void **PCD_Init** (byte chipSelectPin, byte resetPowerDownPin)
- void **PCD_Reset** ()
- void **PCD_AntennaOn** ()
- void **PCD_AntennaOff** ()
- byte **PCD_GetAntennaGain** ()
- void **PCD_SetAntennaGain** (byte mask)
- bool **PCD_PerformSelfTest** ()
- void **PCD_SoftPowerDown** ()
- void **PCD_SoftPowerUp** ()
- StatusCode **PCD_TransceiveData** (byte *sendData, byte sendLen, byte *backData, byte *backLen, byte *validBits=nullptr, byte rxAlign=0, bool checkCRC=false)
- StatusCode **PCD_CommunicateWithPICC** (byte command, byte waitIRq, byte *sendData, byte sendLen, byte *backData=nullptr, byte *backLen=nullptr, byte *validBits=nullptr, byte rxAlign=0, bool checkCRC=false)
- StatusCode **PICC_RequestA** (byte *bufferATQA, byte *bufferSize)
- StatusCode **PICC_WakeupA** (byte *bufferATQA, byte *bufferSize)
- StatusCode **PICC_REQA_or_WUPA** (byte command, byte *bufferATQA, byte *bufferSize)
- virtual StatusCode **PICC_Select** (Uid *uid, byte validBits=0)
- StatusCode **PICC_HaltA** ()
- StatusCode **PCD_Authenticate** (byte command, byte blockAddr, MIFARE_Key *key, Uid *uid)
- void **PCD_StopCrypto1** ()
- StatusCode **MIFARE_Read** (byte blockAddr, byte *buffer, byte *bufferSize)
- StatusCode **MIFARE_Write** (byte blockAddr, byte *buffer, byte bufferSize)
- StatusCode **MIFARE_Ultralight_Write** (byte page, byte *buffer, byte bufferSize)
- StatusCode **MIFARE_Decrement** (byte blockAddr, int32_t delta)
- StatusCode **MIFARE_Increment** (byte blockAddr, int32_t delta)
- StatusCode **MIFARE_Restore** (byte blockAddr)
- StatusCode **MIFARE_Transfer** (byte blockAddr)
- StatusCode **MIFARE_GetValue** (byte blockAddr, int32_t *value)
- StatusCode **MIFARE_SetValue** (byte blockAddr, int32_t value)
- StatusCode **PCD_NTAG216_AUTH** (byte *passWord, byte pACK[])
- StatusCode **PCD_MIFARE_Transceive** (byte *sendData, byte sendLen, bool acceptTimeout=false)
- void **PCD_DumpVersionToSerial** ()
- void **PICC_DumpToSerial** (Uid *uid)
- void **PICC_DumpDetailsToSerial** (Uid *uid)
- void **PICC_DumpMifareClassicToSerial** (Uid *uid, PICC_Type piccType, MIFARE_Key *key)
- void **PICC_DumpMifareClassicSectorToSerial** (Uid *uid, MIFARE_Key *key, byte sector)
- void **PICC_DumpMifareUltralightToSerial** ()
- void **MIFARE_SetAccessBits** (byte *accessBitBuffer, byte g0, byte g1, byte g2, byte g3)
- bool **MIFARE_OpenUidBackdoor** (bool logErrors)
- bool **MIFARE_SetUid** (byte *newUid, byte uidSize, bool logErrors)
- bool **MIFARE_UnbrickUidSector** (bool logErrors)
- virtual bool **PICC_IsNewCardPresent** ()
- virtual bool **PICC_ReadCardSerial** ()

Métodos públicos estáticos

- static const __FlashStringHelper * **GetStatusCodeName** (StatusCode code)
- static PICC_Type **PICC_GetType** (byte sak)
- static const __FlashStringHelper * **PICC_GetTypeName** (PICC_Type type)

Campos de datos

- [Uid](#) uid

Atributos públicos estáticos

- static constexpr byte **FIFO_SIZE** = 64
- static constexpr uint8_t **UNUSED_PIN** = UINT8_MAX

Métodos protegidos

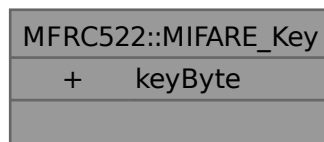
- StatusCode **MIFARE_TwoStepHelper** (byte command, byte blockAddr, int32_t data)

Atributos protegidos

- byte **_chipSelectPin**
- byte **_resetPowerDownPin**

4.3.1. Documentación de estructuras de datos**4.3.1.1. struct MFRC522::MIFARE_Key**

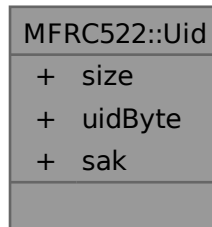
Diagrama de colaboración de MFRC522::MIFARE_Key:

**Campos de datos**

byte	keyByte[MF_KEY_SIZE]	
------	----------------------	--

4.3.1.2. struct MFRC522::Uid

Diagrama de colaboración de MFRC522::Uid:



Campos de datos

byte	sak	
byte	size	
byte	uidByte[10]	

La documentación de esta clase está generada del siguiente archivo:

- main/libraries/RFID-RC522/MFRC522.h

4.4. Referencia de la clase MFRC522Extended

```
#include <MFRC522Extended.h>
```

Diagrama de herencia de MFRC522Extended

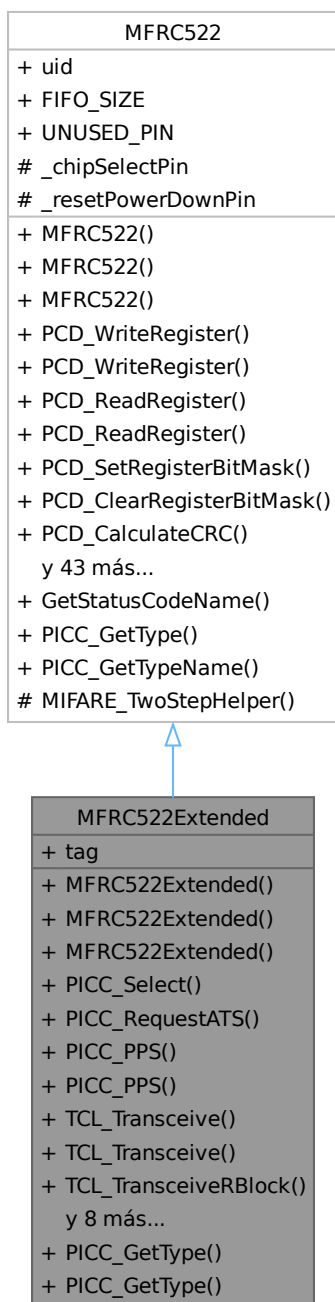
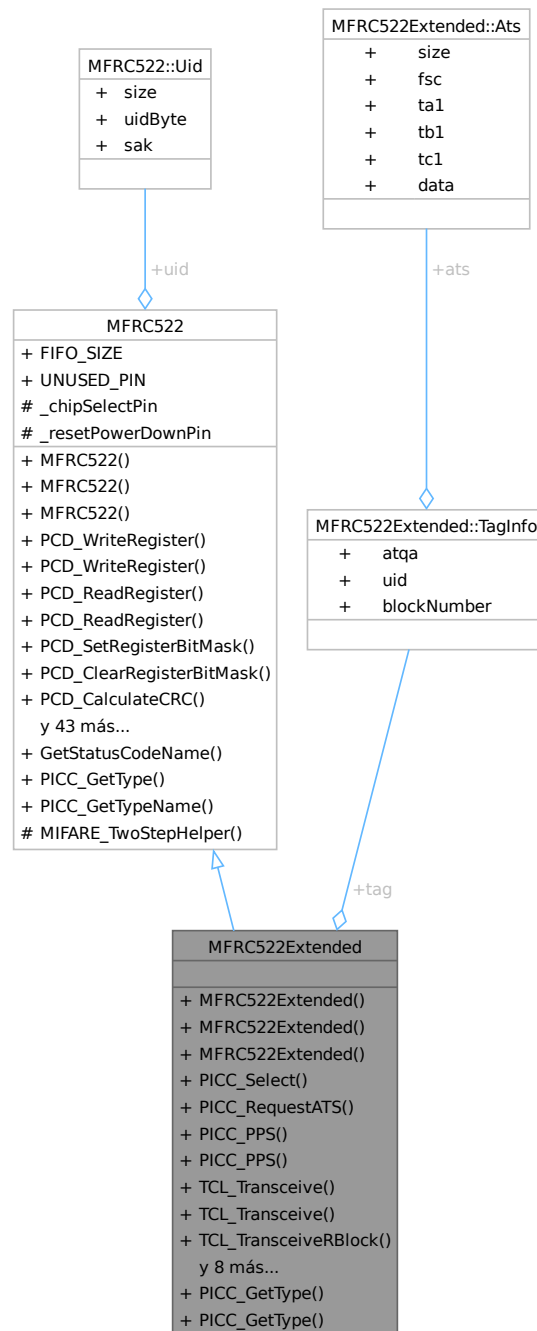


Diagrama de colaboración de MFRC522Extended:



Estructuras de datos

- struct [Ats](#)
- struct [Ats.ta1](#)
- struct [Ats.tb1](#)
- struct [Ats.tc1](#)
- struct [PcbBlock](#)

- struct [PcbBlock.inf](#)
- struct [PcbBlock.prologue](#)
- struct [TagInfo](#)

Tipos públicos

- enum **TagBitRates** : byte { **BITRATE_106KBITS** = 0x00 , **BITRATE_212KBITS** = 0x01 , **BITRATE_424** ← **KBITS** = 0x02 , **BITRATE_848KBITS** = 0x03 }

Tipos públicos heredados de [MFRC522](#)

- enum **PCD_Register** : byte {
CommandReg = 0x01 << 1 , **ComIEnReg** = 0x02 << 1 , **DivIEnReg** = 0x03 << 1 , **ComIrqReg** = 0x04 << 1 ,
DivIrqReg = 0x05 << 1 , **ErrorReg** = 0x06 << 1 , **Status1Reg** = 0x07 << 1 , **Status2Reg** = 0x08 << 1 ,
FIFODataReg = 0x09 << 1 , **FIFOLevelReg** = 0x0A << 1 , **WaterLevelReg** = 0x0B << 1 , **ControlReg** = 0x0C << 1 ,
BitFramingReg = 0x0D << 1 , **CollReg** = 0x0E << 1 , **ModeReg** = 0x11 << 1 , **TxModeReg** = 0x12 << 1 ,
RxModeReg = 0x13 << 1 , **TxControlReg** = 0x14 << 1 , **TxASKReg** = 0x15 << 1 , **TxSelReg** = 0x16 << 1 ,
RxSelReg = 0x17 << 1 , **RxThresholdReg** = 0x18 << 1 , **DemodReg** = 0x19 << 1 , **MfTxReg** = 0x1C << 1 ,
MfRxReg = 0x1D << 1 , **SerialSpeedReg** = 0x1F << 1 , **CRCResultRegH** = 0x21 << 1 , **CRCResultRegL** = 0x22 << 1 ,
ModWidthReg = 0x24 << 1 , **RFCfgReg** = 0x26 << 1 , **GsNReg** = 0x27 << 1 , **CWGsPReg** = 0x28 << 1 ,
ModGsPReg = 0x29 << 1 , **TModeReg** = 0x2A << 1 , **TPrescalerReg** = 0x2B << 1 , **TReloadRegH** = 0x2C << 1 ,
TReloadRegL = 0x2D << 1 , **TCounterValueRegH** = 0x2E << 1 , **TCounterValueRegL** = 0x2F << 1 ,
TestSel1Reg = 0x31 << 1 ,
TestSel2Reg = 0x32 << 1 , **TestPinEnReg** = 0x33 << 1 , **TestPinValueReg** = 0x34 << 1 , **TestBusReg** = 0x35 << 1 ,
AutoTestReg = 0x36 << 1 , **VersionReg** = 0x37 << 1 , **AnalogTestReg** = 0x38 << 1 , **TestDAC1Reg** = 0x39 << 1 ,
TestDAC2Reg = 0x3A << 1 , **TestADCReg** = 0x3B << 1 }
- enum **PCD_Command** : byte {
PCD_Idle = 0x00 , **PCD_Mem** = 0x01 , **PCD_GenerateRandomID** = 0x02 , **PCD_CalcCRC** = 0x03 ,
PCD_Transmit = 0x04 , **PCD_NoCmdChange** = 0x07 , **PCD_Receive** = 0x08 , **PCD_Transceive** = 0x0C ,
PCD_MFAuthent = 0x0E , **PCD_SoftReset** = 0x0F }
- enum **PCD_RxGain** : byte {
RxGain_18dB = 0x00 << 4 , **RxGain_23dB** = 0x01 << 4 , **RxGain_18dB_2** = 0x02 << 4 , **RxGain_** ← **23dB_2** = 0x03 << 4 ,
RxGain_33dB = 0x04 << 4 , **RxGain_38dB** = 0x05 << 4 , **RxGain_43dB** = 0x06 << 4 , **RxGain_48dB** = 0x07 << 4 ,
RxGain_min = 0x00 << 4 , **RxGain_avg** = 0x04 << 4 , **RxGain_max** = 0x07 << 4 }
- enum **PICC_Command** : byte {
PICC_CMD_REQA = 0x26 , **PICC_CMD_WUPA** = 0x52 , **PICC_CMD_CT** = 0x88 , **PICC_CMD_SEL_CL1** = 0x93 ,
PICC_CMD_SEL_CL2 = 0x95 , **PICC_CMD_SEL_CL3** = 0x97 , **PICC_CMD_HLTA** = 0x50 , **PICC_CMD_** ← **RATS** = 0xE0 ,
PICC_CMD_MF_AUTH_KEY_A = 0x60 , **PICC_CMD_MF_AUTH_KEY_B** = 0x61 , **PICC_CMD_MF_READ** = 0x30 , **PICC_CMD_MF_WRITE** = 0xA0 ,
PICC_CMD_MF_DECREMENT = 0xC0 , **PICC_CMD_MF_INCREMENT** = 0xC1 , **PICC_CMD_MF_** ← **RESTORE** = 0xC2 , **PICC_CMD_MF_TRANSFER** = 0xB0 ,
PICC_CMD_UL_WRITE = 0xA2 }

- enum **MIFARE_Misc** { **MF_ACK** = 0xA , **MF_KEY_SIZE** = 6 }
- enum **PICC_Type** : byte {
PICC_TYPE_UNKNOWN , **PICC_TYPE_ISO_14443_4** , **PICC_TYPE_ISO_18092** , **PICC_TYPE_↔**
MIFARE_MINI ,
PICC_TYPE_MIFARE_1K , **PICC_TYPE_MIFARE_4K** , **PICC_TYPE_MIFARE_UL** , **PICC_TYPE_↔**
MIFARE_PLUS ,
PICC_TYPE_MIFARE_DESFIRE , **PICC_TYPE_TNP3XXX** , **PICC_TYPE_NOT_COMPLETE** = 0xff }
- enum **StatusCode** : byte {
STATUS_OK , **STATUS_ERROR** , **STATUS_COLLISION** , **STATUS_TIMEOUT** ,
STATUS_NO_ROOM , **STATUS_INTERNAL_ERROR** , **STATUS_INVALID** , **STATUS_CRC_WRONG** ,
STATUS_MIFARE_NACK = 0xff }

Métodos públicos

- **MFRC522Extended** (uint8_t rst)
- **MFRC522Extended** (uint8_t ss, uint8_t rst)
- StatusCode **PICC_Select** (Uid *uid, byte validBits=0) override
- StatusCode **PICC_RequestATS** (Ats *ats)
- StatusCode **PICC_PPS** ()
- StatusCode **PICC_PPS** (TagBitRates sendBitRate, TagBitRates receiveBitRate)
- StatusCode **TCL_Transceive** (PcbBlock *send, PcbBlock *back)
- StatusCode **TCL_Transceive** (TagInfo *tag, byte *sendData, byte sendLen, byte *backData=NULL, byte *backLen=NULL)
- StatusCode **TCL_TransceiveRBlock** (TagInfo *tag, bool ack, byte *backData=NULL, byte *backLen=NULL)
- StatusCode **TCL_Deselect** (TagInfo *tag)
- void **PICC_DumpToSerial** (TagInfo *tag)
- void **PICC_DumpDetailsToSerial** (TagInfo *tag)
- void **PICC_DumpISO14443_4** (TagInfo *tag)
- bool **PICC_IsNewCardPresent** () override
- bool **PICC_ReadCardSerial** () override
- void **PICC_DumpToSerial** (Uid *uid)
- void **PICC_DumpDetailsToSerial** (Uid *uid)

Métodos públicos heredados de **MFRC522**

- **MFRC522** (byte resetPowerDownPin)
- **MFRC522** (byte chipSelectPin, byte resetPowerDownPin)
- void **PCD_WriteRegister** (PCD_Register reg, byte value)
- void **PCD_WriteRegister** (PCD_Register reg, byte count, byte *values)
- byte **PCD_ReadRegister** (PCD_Register reg)
- void **PCD_ReadRegister** (PCD_Register reg, byte count, byte *values, byte rxAlign=0)
- void **PCD_SetRegisterBitMask** (PCD_Register reg, byte mask)
- void **PCD_ClearRegisterBitMask** (PCD_Register reg, byte mask)
- StatusCode **PCD_CalculateCRC** (byte *data, byte length, byte *result)
- void **PCD_Init** ()
- void **PCD_Init** (byte resetPowerDownPin)
- void **PCD_Init** (byte chipSelectPin, byte resetPowerDownPin)
- void **PCD_Reset** ()
- void **PCD_AntennaOn** ()
- void **PCD_AntennaOff** ()
- byte **PCD_GetAntennaGain** ()
- void **PCD_SetAntennaGain** (byte mask)
- bool **PCD_PerformSelfTest** ()

- void **PCD_SoftPowerDown** ()
- void **PCD_SoftPowerUp** ()
- StatusCode **PCD_TransceiveData** (byte *sendData, byte sendLen, byte *backData, byte *backLen, byte *validBits=nullptr, byte rxAlign=0, bool checkCRC=false)
- StatusCode **PCD_CommunicateWithPICC** (byte command, byte waitIRq, byte *sendData, byte sendLen, byte *backData=nullptr, byte *backLen=nullptr, byte *validBits=nullptr, byte rxAlign=0, bool checkCRC=false)
- StatusCode **PICC_RequestA** (byte *bufferATQA, byte *bufferSize)
- StatusCode **PICC_WakeupA** (byte *bufferATQA, byte *bufferSize)
- StatusCode **PICC_REQA_or_WUPA** (byte command, byte *bufferATQA, byte *bufferSize)
- StatusCode **PICC_HaltA** ()
- StatusCode **PCD_Authenticate** (byte command, byte blockAddr, [MIFARE_Key](#) *key, [Uid](#) *uid)
- void **PCD_StopCrypto1** ()
- StatusCode **MIFARE_Read** (byte blockAddr, byte *buffer, byte *bufferSize)
- StatusCode **MIFARE_Write** (byte blockAddr, byte *buffer, byte bufferSize)
- StatusCode **MIFARE_Ultralight_Write** (byte page, byte *buffer, byte bufferSize)
- StatusCode **MIFARE_Decrement** (byte blockAddr, int32_t delta)
- StatusCode **MIFARE_Increment** (byte blockAddr, int32_t delta)
- StatusCode **MIFARE_Restore** (byte blockAddr)
- StatusCode **MIFARE_Transfer** (byte blockAddr)
- StatusCode **MIFARE_GetValue** (byte blockAddr, int32_t *value)
- StatusCode **MIFARE_SetValue** (byte blockAddr, int32_t value)
- StatusCode **PCD_NTAG216_AUTH** (byte *passWord, byte pACK[])
- StatusCode **PCD_MIFARE_Transceive** (byte *sendData, byte sendLen, bool acceptTimeout=false)
- void **PCD_DumpVersionToSerial** ()
- void **PICC_DumpToSerial** ([Uid](#) *uid)
- void **PICC_DumpDetailsToSerial** ([Uid](#) *uid)
- void **PICC_DumpMifareClassicToSerial** ([Uid](#) *uid, PICC_Type piccType, [MIFARE_Key](#) *key)
- void **PICC_DumpMifareClassicSectorToSerial** ([Uid](#) *uid, [MIFARE_Key](#) *key, byte sector)
- void **PICC_DumpMifareUltralightToSerial** ()
- void **MIFARE_SetAccessBits** (byte *accessBitBuffer, byte g0, byte g1, byte g2, byte g3)
- bool **MIFARE_OpenUidBackdoor** (bool logErrors)
- bool **MIFARE_SetUid** (byte *newUid, byte uidSize, bool logErrors)
- bool **MIFARE_UnbrickUidSector** (bool logErrors)

Métodos públicos estáticos

- static PICC_Type **PICC_GetType** ([TagInfo](#) *tag)
- static PICC_Type **PICC_GetType** (byte sak)

Métodos públicos estáticos heredados de [MFRC522](#)

- static const __FlashStringHelper * **GetStatusCodeName** (StatusCode code)
- static PICC_Type **PICC_GetType** (byte sak)
- static const __FlashStringHelper * **PICC_GetTypeName** (PICC_Type type)

Campos de datos

- [TagInfo](#) tag

Campos de datos heredados de [MFRC522](#)

- [Uid](#) uid

Otros miembros heredados

Atributos públicos estáticos heredados de [MFRC522](#)

- static constexpr byte **FIFO_SIZE** = 64
- static constexpr uint8_t **UNUSED_PIN** = UINT8_MAX

Métodos protegidos heredados de [MFRC522](#)

- StatusCode **MIFARE_TwoStepHelper** (byte command, byte blockAddr, int32_t data)

Atributos protegidos heredados de [MFRC522](#)

- byte **_chipSelectPin**
- byte **_resetPowerDownPin**

4.4.1. Descripción detallada

Library extends [MFRC522.h](#) to support RATS for ISO-14443-4 PICC. RATS - Request for Answer To Select.

Autor

JPG-Consulting

4.4.2. Documentación de estructuras de datos

4.4.2.1. struct MFRC522Extended::Ats

Diagrama de colaboración de MFRC522Extended::Ats:

MFRC522Extended::Ats	
+	size
+	fsc
+	ta1
+	tb1
+	tc1
+	data

Campos de datos

byte	data[FIFO_SIZE - 2]	
byte	fsc	
byte	size	
struct Ats.ta1	ta1	
struct Ats.tb1	tb1	
struct Ats.tc1	tc1	

4.4.2.2. struct MFRC522Extended::Ats.ta1

Diagrama de colaboración de MFRC522Extended::Ats.ta1:

MFRC522Extended::Ats.ta1	
+	transmitted
+	sameD
+	ds
+	dr

Campos de datos

TagBitRates	dr	
TagBitRates	ds	
bool	sameD	
bool	transmitted	

4.4.2.3. struct MFRC522Extended::Ats.tb1

Diagrama de colaboración de MFRC522Extended::Ats.tb1:

MFRC522Extended::Ats.tb1	
+	transmitted
+	fwi
+	sfgi

Campos de datos

byte	fwi	
byte	sfgi	
bool	transmitted	

4.4.2.4. struct MFRC522Extended::Ats.tc1

Diagrama de colaboración de MFRC522Extended::Ats.tc1:

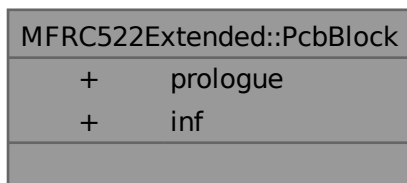
MFRC522Extended::Ats.tc1	
+	transmitted
+	supportsCID
+	supportsNAD

Campos de datos

bool	supportsCID	
bool	supportsNAD	
bool	transmitted	

4.4.2.5. struct MFRC522Extended::PcbBlock

Diagrama de colaboración de MFRC522Extended::PcbBlock:

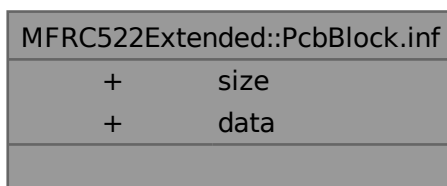


Campos de datos

struct PcbBlock.inf	inf	
struct PcbBlock.prologue	prologue	

4.4.2.6. struct MFRC522Extended::PcbBlock.inf

Diagrama de colaboración de MFRC522Extended::PcbBlock.inf:



Campos de datos

byte *	data	
byte	size	

4.4.2.7. struct MFRC522Extended::PcbBlock.prologue

Diagrama de colaboración de MFRC522Extended::PcbBlock.prologue:

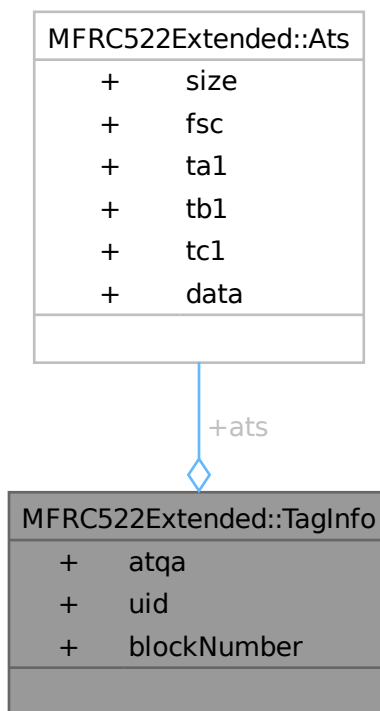
MFRC522Extended::PcbBlock.prologue		
+	pcb	
+	cid	
+	nad	

Campos de datos

byte	cid	
byte	nad	
byte	pcb	

4.4.2.8. struct MFRC522Extended::TagInfo

Diagrama de colaboración de MFRC522Extended::TagInfo:



Campos de datos

uint16_t	atqa	
Ats	ats	
bool	blockNumber	
Uid	uid	

4.4.3. Documentación de funciones miembro

4.4.3.1. PICC_IsNewCardPresent()

```
bool MFRC522Extended::PICC_IsNewCardPresent () [override], [virtual]
```

Reimplementado de [MFRC522](#).

4.4.3.2. PICC_ReadCardSerial()

```
bool MFRC522Extended::PICC_ReadCardSerial () [override], [virtual]
```

Reimplementado de [MFRC522](#).

4.4.3.3. `PICC_Select()`

```
StatusCode MFRC522Extended::PICC_Select (
    Uid * uid,
    byte validBits = 0) [override], [virtual]
```

Reimplementado de [MFRC522](#).

La documentación de esta clase está generada del siguiente archivo:

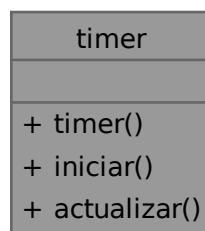
- `main/libraries/RFID-RC522/MFRC522Extended.h`

4.5. Referencia de la clase timer

Clase para gestionar un temporizador con dos salidas, cada una es activada en intervalos distintos, con la funcionalidad de antirrebote en los pines de ingreso y reset.

```
#include <timer.h>
```

Diagrama de colaboración de timer:



Métodos públicos

- **timer** (int pIngreso, int pReset, int p1, int p2, unsigned long t1, unsigned long t2, unsigned long rebote)
Constructor de la clase timer.
- void **iniciar** ()
void [iniciar\(\)](#): configura los pines del objeto de clase timer como entradas y salidas.
- void **actualizar** ()
void [actualizar\(\)](#): Actualiza el estado del temporizador y controla las salidas según los tiempos establecidos. Verifica el estado de los pines de ingreso y reset, aplicando antirrebote.

4.5.1. Descripción detallada

Clase para gestionar un temporizador con dos salidas, cada una es activada en intervalos distintos, con la funcionalidad de antirrebote en los pines de ingreso y reset.

4.5.2. Documentación de constructores y destructores

4.5.2.1. timer()

```
timer::timer (
    int  pIngreso,
    int  pReset,
    int  p1,
    int  p2,
    unsigned long t1,
    unsigned long t2,
    unsigned long rebote)
```

Constructor de la clase timer.

Parámetros

<i>pIngreso</i>	Pin de ingreso para activar el temporizador.
<i>pReset</i>	Pin para resetear el temporizador.
<i>p1</i>	Pin de la primera salida, activado después de tiempoUno.
<i>p2</i>	Pin de la segunda salida, activado después de tiempoDos.
<i>t1</i>	Tiempo (en milisegundos) después para activar la primera salida.
<i>t2</i>	Tiempo (en milisegundos) después para activar la segunda salida.
<i>rebote</i>	Tiempo de antirrebote para los pines de ingreso y reset.

La documentación de esta clase está generada del siguiente archivo:

- main/libraries/TIMER/[timer.h](#)

Capítulo 5

Documentación de archivos

5.1. DHT11.h

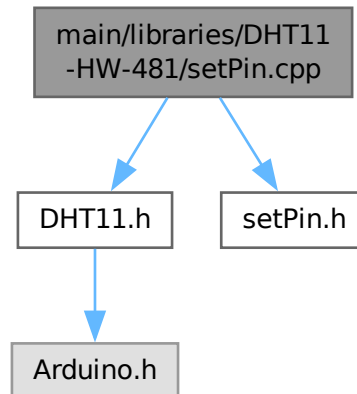
```
00001
00011 #ifndef DHT11_h
00012 #define DHT11_h
00013
00014 #include "Arduino.h"
00015
00021 class DHT11
00022 {
00023
00024 public:
00031     DHT11(int pin);
00032
00039     void setDelay(unsigned long delay);
00040
00047     int readHumidity();
00048
00055     int readTemperature();
00056
00064     int readTemperatureHumidity(int &temperature, int &humidity);
00065
00066     // Constants to represent error codes.
00067     static const int ERROR_CHECKSUM = 254;    // Error code indicating checksum mismatch.
00068     static const int ERROR_TIMEOUT = 253;    // Error code indicating a timeout occurred during
reading.
00069     static const int TIMEOUT_DURATION = 1000; // Duration (in milliseconds) to wait before timing out.
00070
00077     // static String getErrorString(int errorCode);
00078
00079 private:
00080     int _pin;                // Pin number used for communication with the DHT11 sensor.
00081     unsigned long _delayMS = 500; // Default delay in milliseconds between sensor readings.
00082
00093     int readRawData(byte data[5]);
00094
00100     byte readByte();
00101
00107     void startSignal();
00108
00109     friend void setPinDHT11(DHT11& sensor, int pin);
00110
00111 };
00112
00113 #endif
```

5.2. Referencia del archivo main/libraries/DHT11-HW-481/setPin.cpp

Función amiga setPinDHT11.

```
#include "DHT11.h"
#include "setPin.h"
```

Gráfico de dependencias incluidas en setPin.cpp:



Funciones

- void `setPinDHT11` (`DHT11` &sensor, int pin)

5.2.1. Descripción detallada

Función amiga `setPinDHT11`.

Parámetros

<i>DHT11</i> &	sensor: recibe una referencia al objeto de la clase <code>DHT11</code> al cual se le cambiará el pin de conexión correspondiente al sensor de temperatura.
<i>int</i>	pin: es el nuevo pin para la entrada de datos del sensor de temperatura.

La función comprueba si el pin es válido, luego "desactiva" el pin anterior para inicializar el nuevo.

5.2.2. Documentación de funciones

5.2.2.1. `setPinDHT11()`

```
void setPinDHT11 (
    DHT11 & sensor,
    int pin)
```

Bloque if-else Controla si el pin ingresado es uno físicamente válido. Imprime por puerto serie UART un mensaje de error en caso de no serlo.

`pinMode(sensor._pin, INPUT)` "Desactiva" el pin anterior.

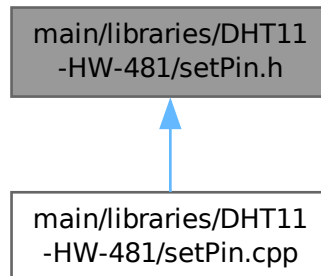
`sensor._pin = pin` Actualiza el pin.

`pinMode(pin, OUTPUT)` y `digitalWrite(pin, HIGH)` Inicializa el nuevo pin como lo hace internamente la clase `DHT11`

`delay(250)` Tiempo de espera a que el sensor se estabilice.

5.3. Referencia del archivo main/libraries/DHT11-HW-481/setPin.h

Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Funciones

- void `setPinDHT11` (`DHT11` &sensor, int pin)

5.3.1. Descripción detallada

Este archivo contiene las declaraciones de la clase `DHT11` y su función amiga `setPinDHT11`.

5.3.2. Documentación de funciones

5.3.2.1. `setPinDHT11()`

```
void setPinDHT11 (  
    DHT11 & sensor,  
    int pin)
```

Bloque if-else Controla si el pin ingresado es uno físicamente válido. Imprime por puerto serie UART un mensaje de error en caso de no serlo.

`pinMode(sensor._pin, INPUT)` "Desactiva" el pin anterior.

`sensor._pin = pin` Actualiza el pin.

`pinMode(pin, OUTPUT)` y `digitalWrite(pin, HIGH)` Inicializa el nuevo pin como lo hace internamente la clase `DHT11`

`delay(250)` Tiempo de espera a que el sensor se estabilice.

5.4. setPin.h

[Ir a la documentación de este archivo.](#)

```
00001 #ifndef SETPIN_H
00002 #define SETPIN_H
00003
00009 class DHT11;
00010 void setPinDHT11(DHT11& sensor, int pin);
00011
00012 #endif
```

5.5. accesoRFID.h

```
00001 #ifndef ACCESORRFID_H
00002 #define ACCESORRFID_H
00003
00004 #include <SPI.h>
00005 #include "MFRC522.h"
00006
00016 class accesoRFID {
00017     public:
00018
00027         accesoRFID(byte ssPin, byte rstPin, String autorizarID);
00028
00034         void start();
00035
00042         void mostrarID();
00043
00049         bool autorizar();
00050
00056         bool detectarTarjeta();
00057
00058     private:
00059         MFRC522 rdif;
00060         String autorizarID;
00061
00062 };
00063
00064 #endif
```

5.6. deprecated.h

```
00001
00007 #ifndef DEPRECATED_H
00008 #define DEPRECATED_H
00009
00010 #ifdef __has_cpp_attribute
00011 #if __has_cpp_attribute(deprecated)
00012 #define DEPRECATED [[deprecated]]
00013 #define DEPRECATED_MSG(msg) [[deprecated(msg)]]
00014 #endif // __has_cpp_attribute(deprecated)
00015 #else
00016 #define DEPRECATED __attribute__((deprecated))
00017 #define DEPRECATED_MSG(msg) __attribute__((deprecated(msg)))
00018 #endif // __has_cpp_attribute
00019
00020 #endif // DEPRECATED_H
```

5.7. MFRC522.h

```
00001
00010 #ifndef MFRC522_h
00011 #define MFRC522_h
00012
00013 #include "require_cpp11.h"
00014 #include "deprecated.h"
00015 // Enable integer limits
00016 #define __STDC_LIMIT_MACROS
00017 #include <stdint.h>
00018 #include <Arduino.h>
00019 #include <SPI.h>
00020
00021 #ifndef MFRC522_SPICLOCK
```

```

00022 #define MFRC522_SPICLOCK (4000000u) // MFRC522 accept upto 10MHz, set to 4MHz.
00023 #endif
00024
00025 // Firmware data for self-test
00026 // Reference values based on firmware version
00027 // Hint: if needed, you can remove unused self-test data to save flash memory
00028 //
00029 // Version 0.0 (0x90)
00030 // Philips Semiconductors; Preliminary Specification Revision 2.0 - 01 August 2005; 16.1 self-test
00031 const byte MFRC522_firmware_referenceV0_0[] PROGMEM = {
00032     0x00, 0x87, 0x98, 0x0F, 0x49, 0xFF, 0x07, 0x19,
00033     0xBF, 0x22, 0x30, 0x49, 0x59, 0x63, 0xAD, 0xCA,
00034     0x7F, 0xE3, 0x4E, 0x03, 0x5C, 0x4E, 0x49, 0x50,
00035     0x47, 0x9A, 0x37, 0x61, 0xE7, 0xE2, 0xC6, 0x2E,
00036     0x75, 0x5A, 0xED, 0x04, 0x3D, 0x02, 0x4B, 0x78,
00037     0x32, 0xFF, 0x58, 0x3B, 0x7C, 0xE9, 0x00, 0x94,
00038     0xB4, 0x4A, 0x59, 0x5B, 0xFD, 0xC9, 0x29, 0xDF,
00039     0x35, 0x96, 0x98, 0x9E, 0x4F, 0x30, 0x32, 0x8D
00040 };
00041 // Version 1.0 (0x91)
00042 // NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
00043 const byte MFRC522_firmware_referenceV1_0[] PROGMEM = {
00044     0x00, 0xC6, 0x37, 0xD5, 0x32, 0xB7, 0x57, 0x5C,
00045     0xC2, 0xD8, 0x7C, 0x4D, 0xD9, 0x70, 0xC7, 0x73,
00046     0x10, 0xE6, 0xD2, 0xAA, 0x5E, 0xA1, 0x3E, 0x5A,
00047     0x14, 0xAF, 0x30, 0x61, 0xC9, 0x70, 0xDB, 0x2E,
00048     0x64, 0x22, 0x72, 0xB5, 0xBD, 0x65, 0xF4, 0xEC,
00049     0x22, 0xBC, 0xD3, 0x72, 0x35, 0xCD, 0xAA, 0x41,
00050     0x1F, 0xA7, 0xF3, 0x53, 0x14, 0xDE, 0x7E, 0x02,
00051     0xD9, 0x0F, 0xB5, 0x5E, 0x25, 0x1D, 0x29, 0x79
00052 };
00053 // Version 2.0 (0x92)
00054 // NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
00055 const byte MFRC522_firmware_referenceV2_0[] PROGMEM = {
00056     0x00, 0xEB, 0x66, 0xBA, 0x57, 0xBF, 0x23, 0x95,
00057     0xD0, 0xE3, 0x0D, 0x3D, 0x27, 0x89, 0x5C, 0xDE,
00058     0x9D, 0x3B, 0xA7, 0x00, 0x21, 0x5B, 0x89, 0x82,
00059     0x51, 0x3A, 0xEB, 0x02, 0x0C, 0xA5, 0x00, 0x49,
00060     0x7C, 0x84, 0x4D, 0xB3, 0xCC, 0xD2, 0x1B, 0x81,
00061     0x5D, 0x48, 0x76, 0xD5, 0x71, 0x61, 0x21, 0xA9,
00062     0x86, 0x96, 0x83, 0x38, 0xCF, 0x9D, 0x5B, 0x6D,
00063     0xDC, 0x15, 0xBA, 0x3E, 0x7D, 0x95, 0x3B, 0x2F
00064 };
00065 // Clone
00066 // Fudan Semiconductor FM17522 (0x88)
00067 const byte FM17522_firmware_reference[] PROGMEM = {
00068     0x00, 0xD6, 0x78, 0x8C, 0xE2, 0xAA, 0x0C, 0x18,
00069     0x2A, 0xB8, 0x7A, 0x7F, 0xD3, 0x6A, 0xCF, 0x0B,
00070     0xB1, 0x37, 0x63, 0x4B, 0x69, 0xAE, 0x91, 0xC7,
00071     0xC3, 0x97, 0xAE, 0x77, 0xF4, 0x37, 0xD7, 0x9B,
00072     0x7C, 0xF5, 0x3C, 0x11, 0x8F, 0x15, 0xC3, 0xD7,
00073     0xC1, 0x5B, 0x00, 0x2A, 0xD0, 0x75, 0xDE, 0x9E,
00074     0x51, 0x64, 0xAB, 0x3E, 0xE9, 0x15, 0xB5, 0xAB,
00075     0x56, 0x9A, 0x98, 0x82, 0x26, 0xEA, 0x2A, 0x62
00076 };
00077
00078 class MFRC522 {
00079 public:
00080     // Size of the MFRC522 FIFO
00081     static constexpr byte FIFO_SIZE = 64; // The FIFO is 64 bytes.
00082     // Default value for unused pin
00083     static constexpr uint8_t UNUSED_PIN = UINT8_MAX;
00084
00085     // MFRC522 registers. Described in chapter 9 of the datasheet.
00086     // When using SPI all addresses are shifted one bit left in the "SPI address byte" (section
00087     8.1.2.3)
00088     enum PCD_Register : byte {
00089         // Page 0: Command and status
00090         //
00091         CommandReg          = 0x01 << 1, // starts and stops command execution
00092         ComIEnReg           = 0x02 << 1, // enable and disable interrupt request control bits
00093         DivIEnReg           = 0x03 << 1, // enable and disable interrupt request control bits
00094         ComIrqReg           = 0x04 << 1, // interrupt request bits
00095         DivIrqReg           = 0x05 << 1, // interrupt request bits
00096         ErrorReg            = 0x06 << 1, // error bits showing the error status of the last
00097         command executed
00098         Status1Reg          = 0x07 << 1, // communication status bits
00099         Status2Reg          = 0x08 << 1, // receiver and transmitter status bits
00100         FIFODataReg         = 0x09 << 1, // input and output of 64 byte FIFO buffer
00101         FIFOLevelReg        = 0x0A << 1, // number of bytes stored in the FIFO buffer
00102         WaterLevelReg       = 0x0B << 1, // level for FIFO underflow and overflow warning
00103         ControlReg          = 0x0C << 1, // miscellaneous control registers
00104         BitFramingReg       = 0x0D << 1, // adjustments for bit-oriented frames
00105         CollReg             = 0x0E << 1, // bit position of the first bit-collision detected on
00106         the RF interface
00107         //
00108         //
00109         //
00110         //
00111         //
00112         //
00113         //
00114         //
00115         //
00116         //
00117         //
00118         //
00119         //
00120         //
00121         //
00122         //
00123         //
00124         //
00125         //
00126         //
00127         //
00128         //
00129         //
00130         //
00131         //
00132         //
00133         //
00134         //
00135         //
00136         //
00137         //
00138         //
00139         //
00140         //
00141         //
00142         //
00143         //
00144         //
00145         //
00146         //
00147         //
00148         //
00149         //
00150         //
00151         //
00152         //
00153         //
00154         //
00155         //
00156         //
00157         //
00158         //
00159         //
00160         //
00161         //
00162         //
00163         //
00164         //
00165         //
00166         //
00167         //
00168         //
00169         //
00170         //
00171         //
00172         //
00173         //
00174         //
00175         //
00176         //
00177         //
00178         //
00179         //
00180         //
00181         //
00182         //
00183         //
00184         //
00185         //
00186         //
00187         //
00188         //
00189         //
00190         //
00191         //
00192         //
00193         //
00194         //
00195         //
00196         //
00197         //
00198         //
00199         //
00200         //
00201         //
00202         //
00203         //
00204         //
00205         //
00206         //
00207         //
00208         //
00209         //
00210         //
00211         //
00212         //
00213         //
00214         //
00215         //
00216         //
00217         //
00218         //
00219         //
00220         //
00221         //
00222         //
00223         //
00224         //
00225         //
00226         //
00227         //
00228         //
00229         //
00230         //
00231         //
00232         //
00233         //
00234         //
00235         //
00236         //
00237         //
00238         //
00239         //
00240         //
00241         //
00242         //
00243         //
00244         //
00245         //
00246         //
00247         //
00248         //
00249         //
00250         //
00251         //
00252         //
00253         //
00254         //
00255         //
00256         //
00257         //
00258         //
00259         //
00260         //
00261         //
00262         //
00263         //
00264         //
00265         //
00266         //
00267         //
00268         //
00269         //
00270         //
00271         //
00272         //
00273         //
00274         //
00275         //
00276         //
00277         //
00278         //
00279         //
00280         //
00281         //
00282         //
00283         //
00284         //
00285         //
00286         //
00287         //
00288         //
00289         //
00290         //
00291         //
00292         //
00293         //
00294         //
00295         //
00296         //
00297         //
00298         //
00299         //
00300         //
00301         //
00302         //
00303         //
00304         //
00305         //
00306         //
00307         //
00308         //
00309         //
00310         //
00311         //
00312         //
00313         //
00314         //
00315         //
00316         //
00317         //
00318         //
00319         //
00320         //
00321         //
00322         //
00323         //
00324         //
00325         //
00326         //
00327         //
00328         //
00329         //
00330         //
00331         //
00332         //
00333         //
00334         //
00335         //
00336         //
00337         //
00338         //
00339         //
00340         //
00341         //
00342         //
00343         //
00344         //
00345         //
00346         //
00347         //
00348         //
00349         //
00350         //
00351         //
00352         //
00353         //
00354         //
00355         //
00356         //
00357         //
00358         //
00359         //
00360         //
00361         //
00362         //
00363         //
00364         //
00365         //
00366         //
00367         //
00368         //
00369         //
00370         //
00371         //
00372         //
00373         //
00374         //
00375         //
00376         //
00377         //
00378         //
00379         //
00380         //
00381         //
00382         //
00383         //
00384         //
00385         //
00386         //
00387         //
00388         //
00389         //
00390         //
00391         //
00392         //
00393         //
00394         //
00395         //
00396         //
00397         //
00398         //
00399         //
00400         //
00401         //
00402         //
00403         //
00404         //
00405         //
00406         //
00407         //
00408         //
00409         //
00410         //
00411         //
00412         //
00413         //
00414         //
00415         //
00416         //
00417         //
00418         //
00419         //
00420         //
00421         //
00422         //
00423         //
00424         //
00425         //
00426         //
00427         //
00428         //
00429         //
00430         //
00431         //
00432         //
00433         //
00434         //
00435         //
00436         //
00437         //
00438         //
00439         //
00440         //
00441         //
00442         //
00443         //
00444         //
00445         //
00446         //
00447         //
00448         //
00449         //
00450         //
00451         //
00452         //
00453         //
00454         //
00455         //
00456         //
00457         //
00458         //
00459         //
00460         //
00461         //
00462         //
00463         //
00464         //
00465         //
00466         //
00467         //
00468         //
00469         //
00470         //
00471         //
00472         //
00473         //
00474         //
00475         //
00476         //
00477         //
00478         //
00479         //
00480         //
00481         //
00482         //
00483         //
00484         //
00485         //
00486         //
00487         //
00488         //
00489         //
00490         //
00491         //
00492         //
00493         //
00494         //
00495         //
00496         //
00497         //
00498         //
00499         //
00500         //
00501         //
00502         //
00503         //
00504         //
00505         //
00506         //
00507         //
00508         //
00509         //
00510         //
00511         //
00512         //
00513         //
00514         //
00515         //
00516         //
00517         //
00518         //
00519         //
00520         //
00521         //
00522         //
00523         //
00524         //
00525         //
00526         //
00527         //
00528         //
00529         //
00530         //
00531         //
00532         //
00533         //
00534         //
00535         //
00536         //
00537         //
00538         //
00539         //
00540         //
00541         //
00542         //
00543         //
00544         //
00545         //
00546         //
00547         //
00548         //
00549         //
00550         //
00551         //
00552         //
00553         //
00554         //
00555         //
00556         //
00557         //
00558         //
00559         //
00560         //
00561         //
00562         //
00563         //
00564         //
00565         //
00566         //
00567         //
00568         //
00569         //
00570         //
00571         //
00572         //
00573         //
00574         //
00575         //
00576         //
00577         //
00578         //
00579         //
00580         //
00581         //
00582         //
00583         //
00584         //
00585         //
00586         //
00587         //
00588         //
00589         //
00590         //
00591         //
00592         //
00593         //
00594         //
00595         //
00596         //
00597         //
00598         //
00599         //
00600         //
00601         //
00602         //
00603         //
00604         //
00605         //
00606         //
00607         //
00608         //
00609         //
00610         //
00611         //
00612         //
00613         //
00614         //
00615         //
00616         //
00617         //
00618         //
00619         //
00620         //
00621         //
00622         //
00623         //
00624         //
00625         //
00626         //
00627         //
00628         //
00629         //
00630         //
00631         //
00632         //
00633         //
00634         //
00635         //
00636         //
00637         //
00638         //
00639         //
00640         //
00641         //
00642         //
00643         //
00644         //
00645         //
00646         //
00647         //
00648         //
00649         //
00650         //
00651         //
00652         //
00653         //
00654         //
00655         //
00656         //
00657         //
00658         //
00659         //
00660         //
00661         //
00662         //
00663         //
00664         //
00665         //
00666         //
00667         //
00668         //
00669         //
00670         //
00671         //
00672         //
00673         //
00674         //
00675         //
00676         //
00677         //
00678         //
00679         //
00680         //
00681         //
00682         //
00683         //
00684         //
00685         //
00686         //
00687         //
00688         //
00689         //
00690         //
00691         //
00692         //
00693         //
00694         //
00695         //
00696         //
00697         //
00698         //
00699         //
00700         //
00701         //
00702         //
00703         //
00704         //
00705         //
00706         //
00707         //
00708         //
00709         //
00710         //
00711         //
00712         //
00713         //
00714         //
00715         //
00716         //
00717         //
00718         //
00719         //
00720         //
00721         //
00722         //
00723         //
00724         //
00725         //
00726         //
00727         //
00728         //
00729         //
00730         //
00731         //
00732         //
00733         //
00734         //
00735         //
00736         //
00737         //
00738         //
00739         //
00740         //
00741         //
00742         //
00743         //
00744         //
00745         //
00746         //
00747         //
00748         //
00749         //
00750         //
00751         //
00752         //
00753         //
00754         //
00755         //
00756         //
00757         //
00758         //
00759         //
00760         //
00761         //
00762         //
00763         //
00764         //
00765         //
00766         //
00767         //
00768         //
00769         //
00770         //
00771         //
00772         //
00773         //
00774         //
00775         //
00776         //
00777         //
00778         //
00779         //
00780         //
00781         //
00782         //
00783         //
00784         //
00785         //
00786         //
00787         //
00788         //
00789         //
00790         //
00791         //
00792         //
00793         //
00794         //
00795         //
00796         //
00797         //
00798         //
00799         //
00800         //
00801         //
00802         //
00803         //
00804         //
00805         //
00806         //
00807         //
00808         //
00809         //
00810         //
00811         //
00812         //
00813         //
00814         //
00815         //
00816         //
00817         //
00818         //
00819         //
00820         //
00821         //
00822         //
00823         //
00824         //
00825         //
00826         //
00827         //
00828         //
00829         //
00830         //
00831         //
00832         //
00833         //
00834         //
00835         //
00836         //
00837         //
00838         //
00839         //
00840         //
00841         //
00842         //
00843         //
00844         //
00845         //
00846         //
00847         //
00848         //
00849         //
00850         //
00851         //
00852         //
00853         //
00854         //
00855         //
00856         //
00857         //
00858         //
00859         //
00860         //
00861         //
00862         //
00863         //
00864         //
00865         //
00866         //
00867         //
00868         //
00869         //
00870         //
00871         //
00872         //
00873         //
00874         //
00875         //
00876         //
00877         //
00878         //
00879         //
00880         //
00881         //
00882         //
00883         //
00884         //
00885         //
00886         //
00887         //
00888         //
00889         //
00890         //
00891         //
00892         //
00893         //
00894         //
00895         //
00896         //
00897         //
00898         //
00899         //
00900         //
00901         //
00902         //
00903         //
00904         //
00905         //
00906         //
00907         //
00908         //
00909         //
00910         //
00911         //
00912         //
00913         //
00914         //
00915         //
00916         //
00917         //
00918         //
00919         //
00920         //
00921         //
00922         //
00923         //
00924         //
00925         //
00926         //
00927         //
00928         //
00929         //
00930         //
00931         //
00932         //
00933         //
00934         //
00935         //
00936         //
00937         //
00938         //
00939         //
00940         //
00941         //
00942         //
00943         //
00944         //
00945         //
00946         //
00947         //
00948         //
00949         //
00950         //
00951         //
00952         //
00953         //
00954         //
00955         //
00956         //
00957         //
00958         //
00959         //
00960         //
00961         //
00962         //
00963         //
00964         //
00965         //
00966         //
00967         //
00968         //
00969         //
00970         //
00971         //
00972         //
00973         //
00974         //
00975         //
00976         //
00977         //
00978         //
00979         //
00980         //
00981         //
00982         //
00983         //
00984         //
00985         //
00986         //
00987         //
00988         //
00989         //
00990         //
00991         //
00992         //
00993         //
00994         //
00995         //
00996         //
00997         //
00998         //
00999         //
01000         //
01001         //
01002         //
01003         //
01004         //
01005         //
01006         //
01007         //
01008         //
01009         //
01010         //
01011         //
01012         //
01013         //
01014         //
01015         //
01016         //
01017         //
01018         //
01019         //
01020         //
01021         //
01022         //
01023         //
01024         //
01025         //
01026         //
01027         //
01028         //
01029         //
01030         //
01031         //
01032         //
01033         //
01034         //
01035         //
01036         //
01037         //
01038         //
01039         //
01040         //
01041         //
01042         //
01043         //
01044         //
01045         //
01046         //
01047         //
01048         //
01049         //
01050         //
01051         //
01052         //
01053         //
01054         //
01055         //
01056         //
01057         //
01058         //
01059         //
01060         //
01061         //
01062         //
01063         //
01064         //
01065         //
01066         //
01067         //
01068         //
01069         //
01070         //
01071         //
01072         //
01073         //
01074         //
01075         //
01076         //
01077         //
01078         //
01079         //
01080         //
01081         //
01082         //
01083         //
01084         //
01085         //
01086         //
01087         //
01088         //
01089         //
01090         //
01091         //
01092         //
01093         //
01094         //
01095         //
01096         //
01097         //
01098         //
01099         //
01100         //
01101         //
01102         //
01103         //
01104         //
01105         //
01106         //
01107         //
01108         //
01109         //
01110         //
01111         //
01112         //
01113         //
01114         //
01115         //
01116         //
01117         //
01118         //
01119         //
01120         //
01121         //
01122         //
01123         //
01124         //
01125         //
01126         //
01127         //
01128         //
01129         //
01130         //
01131         //
01132         //
01133         //
01134         //
01135         //
01136         //
01137         //
01138         //
01139         //
01140         //
01141         //
01142         //
01143         //
01144         //
01145         //
01146         //
01147         //
01148         //
01149         //
01150         //
01151         //
01152         //
01153         //
01154         //
01155         //
01156         //
01157         //
01158         //
01159         //
01160         //
01161         //
01162         //
01163         //
01164         //
01165         //
01166         //
01167         //
01168         //
01169         //
01170         //
01171         //
01172         //
01173         //
01174         //
01175         //
01176         //
01177         //
01178         //
01179         //
01180         //
01181         //
01182         //
01183         //
01184         //
01185         //
01186         //
01187         //
01188         //
01189         //
01190         //
01191         //
01192         //
01193         //
01194         //
01195         //
01196         //
01197         //
01198         //
01199         //
01200         //
01201         //
01202         //
01203         //
01204         //
01205         //
01206         //
01207         //
01208         //
01209         //
01210         //
01211         //
01212         //
01213         //
01214         //
01215         //
01216         //
01217         //
01218         //
01219         //
01220         //
01221         //
01222         //
01223         //
01224         //
01225         //
01226         //
01227         //
01228         //
01229         //
01230         //
01231         //
01232         //
01233         //
01234         //
01235         //
01236         //
01237         //
01238         //
01239         //
01240         //
01241         //
01242         //
01243         //
01244         //
01245         //
01246         //
01247         //
01248         //
01249         //
01250         //
01251         //
01252         //
01253         //
01254         //
01255         //
01256         //
01257         //
01258         //
01259         //
01260         //
01261         //
01262         //
01263         //
01264         //
01265         //
01266         //
01267         //
01268         //
01269         //
01270         //
01271         //
01272         //
01273         //
01274         //
01275         //
01276         //
01277         //
01278         //
01279         //
01280         //
01281         //
01282         //
01283         //
01284         //
01285         //
01286         //
01287         //
01288         //
01289         //
01290         //
01291         //
01292         //
01293         //
01294         //
01295         //
01296         //

```

```

00106 // Page 1: Command
00107 //
00108 ModeReg = 0x10 // reserved for future use
// defines general modes for transmitting and receiving

00109 TxModeReg = 0x12 < 1, // defines transmission data rate and framing
00110 RxModeReg = 0x13 < 1, // defines reception data rate and framing
00111 TxControlReg = 0x14 < 1, // controls the logical behavior of the antenna driver
pins TX1 and TX2
00112 TxASKReg = 0x15 < 1, // controls the setting of the transmission modulation
00113 TxSelReg = 0x16 < 1, // selects the internal sources for the antenna driver
00114 RxSelReg = 0x17 < 1, // selects internal receiver settings
00115 RxThresholdReg = 0x18 < 1, // selects thresholds for the bit decoder
00116 DemodReg = 0x19 < 1, // defines demodulator settings
00117 // 0x1A // reserved for future use
00118 // 0x1B // reserved for future use
00119 MfTxReg = 0x1C < 1, // controls some MIFARE communication transmit
parameters
00120 MfRxReg = 0x1D < 1, // controls some MIFARE communication receive
parameters
00121 // 0x1E // reserved for future use
00122 SerialSpeedReg = 0x1F < 1, // selects the speed of the serial UART interface
00123
00124 // Page 2: Configuration
00125 //
00126 CRCResultRegH = 0x20 // reserved for future use
00127 CRCResultRegL = 0x21 < 1, // shows the MSB and LSB values of the CRC calculation
00128 // 0x22 < 1,
00129 // 0x23 // reserved for future use
00130 ModWidthReg = 0x24 < 1, // controls the ModWidth setting?
00131 // 0x25 // reserved for future use
00132 RFCfgReg = 0x26 < 1, // configures the receiver gain
00133 GsNReg = 0x27 < 1, // selects the conductance of the antenna driver pins
TX1 and TX2 for modulation
00133 CWGsPReg = 0x28 < 1, // defines the conductance of the p-driver output
during periods of no modulation
00134 ModGsPReg = 0x29 < 1, // defines the conductance of the p-driver output
during periods of modulation
00135 TModeReg = 0x2A < 1, // defines settings for the internal timer
00136 TPrescalerReg = 0x2B < 1, // the lower 8 bits of the TPrescaler value. The 4 high
bits are in TModeReg.
00137 TReloadRegH = 0x2C < 1, // defines the 16-bit timer reload value
00138 TReloadRegL = 0x2D < 1,
00139 TCounterValueRegH = 0x2E < 1, // shows the 16-bit timer value
00140 TCounterValueRegL = 0x2F < 1,
00141
00142 // Page 3: Test Registers
00143 //
00144 TestSel1Reg = 0x30 // reserved for future use
00145 TestSel2Reg = 0x31 < 1, // general test signal configuration
00146 TestPinEnReg = 0x32 < 1, // general test signal configuration
00147 TestPinValueReg = 0x33 < 1, // enables pin output driver on pins D1 to D7
an I/O bus
00148 TestBusReg = 0x34 < 1, // defines the values for D1 to D7 when it is used as
00149 TestBusReg = 0x35 < 1, // shows the status of the internal test bus
00150 AutoTestReg = 0x36 < 1, // controls the digital self-test
00151 VersionReg = 0x37 < 1, // controls the pins AUX1 and AUX2
00152 AnalogTestReg = 0x38 < 1, // shows the software version
00153 TestDAC1Reg = 0x39 < 1, // controls the test value for TestDAC1
00154 TestDAC2Reg = 0x3A < 1, // defines the test value for TestDAC2
00155 TestADCReg = 0x3B < 1, // shows the value of ADC I and Q channels
00156 // 0x3C // reserved for production tests
00157 // 0x3D // reserved for production tests
00158 // 0x3E // reserved for production tests
00159 // 0x3F // reserved for production tests
00160
00161 // MFR522 commands. Described in chapter 10 of the datasheet.
00162 enum PCD_Command : byte {
00163 PCD_Idle = 0x00, // no action, cancels current command execution
00164 PCD_Mem = 0x01, // stores 25 bytes into the internal buffer
00165 PCD_GenerateRandomID = 0x02, // generates a 10-byte random ID number
00166 PCD_CalcCRC = 0x03, // activates the CRC coprocessor or performs a self-test
00167 PCD_Transmit = 0x04, // transmits data from the FIFO buffer
00168 PCD_NoCmdChange = 0x07, // no command change, can be used to modify the CommandReg
register bits without affecting the command, for example, the PowerDown bit
00169 PCD_Receive = 0x08, // activates the receiver circuits
00170 PCD_Transceive = 0x0C, // transmits data from FIFO buffer to antenna and
automatically activates the receiver after transmission
00171 PCD_MFAuthent = 0x0E, // performs the MIFARE standard authentication as a reader
00172 PCD_SoftReset = 0x0F, // resets the MFR522
00173 };
00174
00175 // MFR522 RxGain[2:0] masks, defines the receiver's signal voltage gain factor (on the PCD).
00176 // Described in 9.3.3.6 / table 98 of the datasheet at
http://www.nxp.com/documents/data\_sheet/MFR522.pdf
00177 enum PCD_RxGain : byte {
00178 RxGain_18dB = 0x00 < 4, // 000b - 18 dB, minimum
00179 RxGain_23dB = 0x01 < 4, // 001b - 23 dB
00180 RxGain_18dB_2 = 0x02 < 4, // 010b - 18 dB, it seems 010b is a duplicate for 000b

```



```

00181     RxGain_23dB_2      = 0x03 << 4,    // 011b - 23 dB, it seems 011b is a duplicate for 001b
00182     RxGain_33dB       = 0x04 << 4,    // 100b - 33 dB, average, and typical default
00183     RxGain_38dB       = 0x05 << 4,    // 101b - 38 dB
00184     RxGain_43dB       = 0x06 << 4,    // 110b - 43 dB
00185     RxGain_48dB       = 0x07 << 4,    // 111b - 48 dB, maximum
00186     RxGain_min        = 0x00 << 4,    // 000b - 18 dB, minimum, convenience for RxGain_18dB
00187     RxGain_avg        = 0x04 << 4,    // 100b - 33 dB, average, convenience for RxGain_33dB
00188     RxGain_max        = 0x07 << 4,    // 111b - 48 dB, maximum, convenience for RxGain_48dB
00189 };
00190
00191 // Commands sent to the PICC.
00192 enum PICC_Command : byte {
00193     // The commands used by the PCD to manage communication with several PICCs (ISO 14443-3, Type
00194     A, section 6.4)
00195     PICC_CMD_REQA      = 0x26,        // REQuest command, Type A. Invites PICCs in state IDLE to
go to READY and prepare for anticollision or selection. 7 bit frame.
00196     PICC_CMD_WUPA      = 0x52,        // Wake-UP command, Type A. Invites PICCs in state IDLE
and HALT to go to READY(*) and prepare for anticollision or selection. 7 bit frame.
00197     PICC_CMD_CT        = 0x88,        // Cascade Tag. Not really a command, but used during anti
collision.
00198     PICC_CMD_SEL_CL1   = 0x93,        // Anti collision/Select, Cascade Level 1
00199     PICC_CMD_SEL_CL2   = 0x95,        // Anti collision/Select, Cascade Level 2
00200     PICC_CMD_SEL_CL3   = 0x97,        // Anti collision/Select, Cascade Level 3
00201     PICC_CMD_HLTA      = 0x50,        // HaLT command, Type A. Instructs an ACTIVE PICC to go to
state HALT.
00202     PICC_CMD_RATS      = 0xE0,        // Request command for Answer To Reset.
00203     // The commands used for MIFARE Classic (from
http://www.mouser.com/ds/2/302/MF1S503x-89574.pdf, Section 9)
00204     // Use PCD_MFAuthent to authenticate access to a sector, then use these commands to
read/write/modify the blocks on the sector.
00205     // The read/write commands can also be used for MIFARE Ultralight.
00206     PICC_CMD_MF_AUTH_KEY_A = 0x60,    // Perform authentication with Key A
00207     PICC_CMD_MF_AUTH_KEY_B = 0x61,    // Perform authentication with Key B
00208     PICC_CMD_MF_READ      = 0x30,    // Reads one 16 byte block from the authenticated sector
of the PICC. Also used for MIFARE Ultralight.
00209     PICC_CMD_MF_WRITE     = 0xA0,    // Writes one 16 byte block to the authenticated sector of
the PICC. Called "COMPATIBILITY WRITE" for MIFARE Ultralight.
00210     PICC_CMD_MF_DECREMENT = 0xC0,    // Decrements the contents of a block and stores the
result in the internal data register.
00211     PICC_CMD_MF_INCREMENT = 0xC1,    // Increments the contents of a block and stores the
result in the internal data register.
00212     PICC_CMD_MF_RESTORE   = 0xC2,    // Reads the contents of a block into the internal data
register.
00213     PICC_CMD_MF_TRANSFER  = 0xB0,    // Writes the contents of the internal data register to a
block.
00214     // The commands used for MIFARE Ultralight (from
http://www.nxp.com/documents/data\_sheet/MF0ICU1.pdf, Section 8.6)
00215     // The PICC_CMD_MF_READ and PICC_CMD_MF_WRITE can also be used for MIFARE Ultralight.
00216     PICC_CMD_UL_WRITE    = 0xA2,    // Writes one 4 byte page to the PICC.
00217 };
00218 // MIFARE constants that does not fit anywhere else
00219 enum MIFARE_Misc {
00220     MF_ACK              = 0xA,        // The MIFARE Classic uses a 4 bit ACK/NAK. Any other
value than 0xA is NAK.
00221     MF_KEY_SIZE         = 6,          // A Mifare Cryptol key is 6 bytes.
00222 };
00223
00224 // PICC types we can detect. Remember to update PICC_GetTypeName() if you add more.
00225 // last value set to 0xff, then compiler uses less ram, it seems some optimisations are triggered
00226 enum PICC_Type : byte {
00227     PICC_TYPE_UNKNOWN   ,
00228     PICC_TYPE_ISO_14443_4 , // PICC compliant with ISO/IEC 14443-4
00229     PICC_TYPE_ISO_18092 , // PICC compliant with ISO/IEC 18092 (NFC)
00230     PICC_TYPE_MIFARE_MINI , // MIFARE Classic protocol, 320 bytes
00231     PICC_TYPE_MIFARE_1K   , // MIFARE Classic protocol, 1KB
00232     PICC_TYPE_MIFARE_4K   , // MIFARE Classic protocol, 4KB
00233     PICC_TYPE_MIFARE_UL   , // MIFARE Ultralight or Ultralight C
00234     PICC_TYPE_MIFARE_PLUS , // MIFARE Plus
00235     PICC_TYPE_MIFARE_DESFIRE, // MIFARE DESFire
00236     PICC_TYPE_TNP3XXX     , // Only mentioned in NXP AN 10833 MIFARE Type Identification
Procedure
00237     PICC_TYPE_NOT_COMPLETE = 0xff // SAK indicates UID is not complete.
00238 };
00239
00240 // Return codes from the functions in this class. Remember to update GetStatusCodeName() if you
add more.
00241 // last value set to 0xff, then compiler uses less ram, it seems some optimisations are triggered
00242 enum StatusCode : byte {
00243     STATUS_OK           , // Success
00244     STATUS_ERROR        , // Error in communication
00245     STATUS_COLLISION    , // Collision detected
00246     STATUS_TIMEOUT      , // Timeout in communication.
00247     STATUS_NO_ROOM      , // A buffer is not big enough.
00248     STATUS_INTERNAL_ERROR, // Internal error in the code. Should not happen ;-)
00249     STATUS_INVALID      , // Invalid argument.
00250     STATUS_CRC_WRONG    , // The CRC_A does not match

```

```

00251     STATUS_MIFARE_NACK      = 0xff // A MIFARE PICC responded with NAK.
00252 };
00253
00254 // A struct used for passing the UID of a PICC.
00255 typedef struct {
00256     byte    size;           // Number of bytes in the UID. 4, 7 or 10.
00257     byte    uidByte[10];
00258     byte    sak;           // The SAK (Select acknowledge) byte returned from the PICC after
successful selection.
00259 } Uid;
00260
00261 // A struct used for passing a MIFARE Crypto1 key
00262 typedef struct {
00263     byte    keyByte[MF_KEY_SIZE];
00264 } MIFARE_Key;
00265
00266 // Member variables
00267 Uid uid; // Used by PICC_ReadCardSerial().
00268
00269 // Functions for setting up the Arduino
00270 MFRC522();
00271 MFRC522(byte resetPowerDownPin);
00272 MFRC522(byte chipSelectPin, byte resetPowerDownPin);
00273
00274 // Basic interface functions for communicating with the MFRC522
00275 void PCD_WriteRegister(PCD_Register reg, byte value);
00276 void PCD_WriteRegister(PCD_Register reg, byte count, byte *values);
00277 byte PCD_ReadRegister(PCD_Register reg);
00278 void PCD_ReadRegister(PCD_Register reg, byte count, byte *values, byte rxAlign = 0);
00279 void PCD_SetRegisterBitMask(PCD_Register reg, byte mask);
00280 void PCD_ClearRegisterBitMask(PCD_Register reg, byte mask);
00281 StatusCode PCD_CalculateCRC(byte *data, byte length, byte *result);
00282
00283 // Functions for manipulating the MFRC522
00284 void PCD_Init();
00285 void PCD_Init(byte resetPowerDownPin);
00286 void PCD_Init(byte chipSelectPin, byte resetPowerDownPin);
00287 void PCD_Reset();
00288 void PCD_AntennaOn();
00289 void PCD_AntennaOff();
00290 byte PCD_GetAntennaGain();
00291 void PCD_SetAntennaGain(byte mask);
00292 bool PCD_PerformSelfTest();
00293
00294 // Power control functions
00295 void PCD_SoftPowerDown();
00296 void PCD_SoftPowerUp();
00297
00298 // Functions for communicating with PICCs
00299 StatusCode PCD_TransceiveData(byte *sendData, byte sendLen, byte *backData, byte *backLen, byte
*validBits = nullptr, byte rxAlign = 0, bool checkCRC = false);
00300 StatusCode PCD_CommunicateWithPICC(byte command, byte waitIRq, byte *sendData, byte sendLen, byte
*backData = nullptr, byte *backLen = nullptr, byte *validBits = nullptr, byte rxAlign = 0, bool
checkCRC = false);
00301 StatusCode PICC_RequestA(byte *bufferATQA, byte *bufferSize);
00302 StatusCode PICC_WakeupA(byte *bufferATQA, byte *bufferSize);
00303 StatusCode PICC_REQA_or_WUPA(byte command, byte *bufferATQA, byte *bufferSize);
00304 virtual StatusCode PICC_Select(Uid *uid, byte validBits = 0);
00305 StatusCode PICC_HaltA();
00306
00307 // Functions for communicating with MIFARE PICCs
00308 StatusCode PCD_Authenticate(byte command, byte blockAddr, MIFARE_Key *key, Uid *uid);
00309 void PCD_StopCrypto1();
00310 StatusCode MIFARE_Read(byte blockAddr, byte *buffer, byte *bufferSize);
00311 StatusCode MIFARE_Write(byte blockAddr, byte *buffer, byte *bufferSize);
00312 StatusCode MIFARE_Ultralight_Write(byte page, byte *buffer, byte *bufferSize);
00313 StatusCode MIFARE_Decrement(byte blockAddr, int32_t delta);
00314 StatusCode MIFARE_Increment(byte blockAddr, int32_t delta);
00315 StatusCode MIFARE_Restore(byte blockAddr);
00316 StatusCode MIFARE_Transfer(byte blockAddr);
00317 StatusCode MIFARE_GetValue(byte blockAddr, int32_t *value);
00318 StatusCode MIFARE_SetValue(byte blockAddr, int32_t value);
00319 StatusCode PCD_NTAG216_AUTH(byte *passWord, byte pACK[]);
00320
00321 // Support functions
00322 StatusCode PCD_MIFARE_Transceive(byte *sendData, byte sendLen, bool acceptTimeout = false);
00323 // old function used too much memory, now name moved to flash; if you need char, copy from flash
to memory
00324 //const char *GetStatusCodeName(byte code);
00325 static const __FlashStringHelper *GetStatusCodeName(StatusCode code);
00326 static PICC_Type PICC_GetType(byte sak);
00327 // old function used too much memory, now name moved to flash; if you need char, copy from flash
to memory
00328 //const char *PICC_GetTypeName(byte type);
00329 static const __FlashStringHelper *PICC_GetTypeName(PICC_Type type);
00330
00331 // Support functions for debugging

```

```

00346 void PCD_DumpVersionToSerial();
00347 void PICC_DumpToSerial(Uid *uid);
00348 void PICC_DumpDetailsToSerial(Uid *uid);
00349 void PICC_DumpMifareClassicToSerial(Uid *uid, PICC_Type piccType, MIFARE_Key *key);
00350 void PICC_DumpMifareClassicSectorToSerial(Uid *uid, MIFARE_Key *key, byte sector);
00351 void PICC_DumpMifareUltralightToSerial();
00352
00353 // Advanced functions for MIFARE
00354 void MIFARE_SetAccessBits(byte *accessBitBuffer, byte g0, byte g1, byte g2, byte g3);
00355 bool MIFARE_OpenUidBackdoor(bool logErrors);
00356 bool MIFARE_SetUid(byte *newUid, byte uidSize, bool logErrors);
00357 bool MIFARE_UnbrickUidSector(bool logErrors);
00358
00360 // Convenience functions - does not add extra functionality
00362 virtual bool PICC_IsNewCardPresent();
00363 virtual bool PICC_ReadCardSerial();
00364
00365 protected:
00366 byte _chipSelectPin; // Arduino pin connected to MFRC522's SPI slave select input (Pin 24,
NSS, active low)
00367 byte _resetPowerDownPin; // Arduino pin connected to MFRC522's reset and power down input (Pin
6, NRSTPD, active low)
00368 StatusCode MIFARE_TwoStepHelper(byte command, byte blockAddr, int32_t data);
00369 };
00370
00371 #endif

```

5.8. MFRC522Extended.h

```

00001
00006 #ifndef MFRC522Extended_h
00007 #define MFRC522Extended_h
00008
00009 #include <Arduino.h>
00010 #include "MFRC522.h"
00011
00012 class MFRC522Extended : public MFRC522 {
00013
00014 public:
00015 // ISO/IEC 14443-4 bit rates
00016 enum TagBitRates : byte {
00017     BITRATE_106KBITS = 0x00,
00018     BITRATE_212KBITS = 0x01,
00019     BITRATE_424KBITS = 0x02,
00020     BITRATE_848KBITS = 0x03
00021 };
00022
00023 // Structure to store ISO/IEC 14443-4 ATS
00024 typedef struct {
00025     byte size;
00026     byte fsc; // Frame size for proximity card
00027
00028     struct {
00029         bool transmitted;
00030         bool sameD; // Only the same D for both directions supported
00031         TagBitRates ds; // Send D
00032         TagBitRates dr; // Receive D
00033     } tdl;
00034
00035     struct {
00036         bool transmitted;
00037         byte fwi; // Frame waiting time integer
00038         byte sfgi; // Start-up frame guard time integer
00039     } tbi;
00040
00041     struct {
00042         bool transmitted;
00043         bool supportsCID;
00044         bool supportsNAD;
00045     } tci;
00046
00047 // Raw data from ATS
00048 byte data[FIFO_SIZE - 2]; // ATS cannot be bigger than FSD - 2 bytes (CRC), according to ISO
14443-4 5.2.2
00049 } Ats;
00050
00051 // A struct used for passing the PICC information
00052 typedef struct {
00053     uint16_t atqa;
00054     Uid uid;
00055     Ats ats;
00056
00057 // For Block PCB

```

```

00058         bool blockNumber;
00059     } TagInfo;
00060
00061     // A struct used for passing PCB Block
00062     typedef struct {
00063         struct {
00064             byte pcb;
00065             byte cid;
00066             byte nad;
00067         } prologue;
00068         struct {
00069             byte size;
00070             byte *data;
00071         } inf;
00072     } PcbBlock;
00073
00074     // Member variables
00075     TagInfo tag;
00076
00077     // Constructors
00078     MFRC522Extended() : MFRC522() {};
00079     MFRC522Extended(uint8_t rst) : MFRC522(rst) {};
00080     MFRC522Extended(uint8_t ss, uint8_t rst) : MFRC522(ss, rst) {};
00081
00082     // Functions for communicating with PICCs
00083     StatusCode PICC_Select(Uid *uid, byte validBits = 0) override; // override
00084     StatusCode PICC_RequestATS(Ats *ats);
00085     StatusCode PICC_PPS(); // PPS command without
00086     bitrate parameter
00087     StatusCode PICC_PPS(TagBitRates sendBitRate, TagBitRates receiveBitRate); // Different D values
00088
00089     // Functions for communicating with ISO/IEC 14433-4 cards
00090     StatusCode TCL_Transceive(PcbBlock *send, PcbBlock *back);
00091     StatusCode TCL_Transceive(TagInfo * tag, byte *sendData, byte sendLen, byte *backData = NULL, byte
00092     *backLen = NULL);
00093     StatusCode TCL_TransceiveRBlock(TagInfo *tag, bool ack, byte *backData = NULL, byte *backLen =
00094     NULL);
00095     StatusCode TCL_Deselect(TagInfo *tag);
00096
00097     // Support functions
00098     static PICC_Type PICC_GetType(TagInfo *tag);
00099     using MFRC522::PICC_GetType; // make old PICC_GetType(byte sak) available, otherwise would be
00100     hidden by PICC_GetType(TagInfo *tag)
00101
00102     // Support functions for debugging
00103     void PICC_DumpToSerial(TagInfo *tag);
00104     using MFRC522::PICC_DumpToSerial; // make old PICC_DumpToSerial(Uid *uid) available, otherwise
00105     would be hidden by PICC_DumpToSerial(TagInfo *tag)
00106     void PICC_DumpDetailsToSerial(TagInfo *tag);
00107     using MFRC522::PICC_DumpDetailsToSerial; // make old PICC_DumpDetailsToSerial(Uid *uid) available,
00108     otherwise would be hidden by PICC_DumpDetailsToSerial(TagInfo *tag)
00109     void PICC_DumpISO14443_4(TagInfo *tag);
00110
00111     // Convenience functions - does not add extra functionality
00112     bool PICC_IsNewCardPresent() override; // override
00113     bool PICC_ReadCardSerial() override; // override
00114 };
00115
00116 #endif

```

5.9. require_cpp11.h

```

00001
00002 #ifndef REQUIRE_CPP11_H
00003 #define REQUIRE_CPP11_H
00004
00005 #if __cplusplus < 201103L
00006 #error "This library needs at least a C++11 compliant compiler, maybe compiler argument for C++11
00007 support is missing or if you use Arduino IDE upgrade to version >=1.6.6"
00008 #endif
00009
00010 #endif
00011
00012 #endif // REQUIRE_CPP11_H

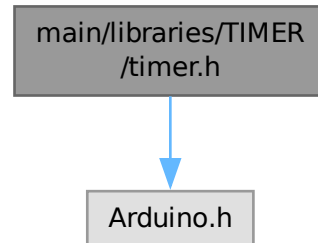
```

5.10. Referencia del archivo main/libraries/TIMER/timer.h

Declaración de la clase timer para manejar dos fases de un temporizador, con antirrebote y un reset.

```
#include <Arduino.h>
```

Gráfico de dependencias incluidas en timer.h:



Estructuras de datos

- class **timer**

Clase para gestionar un temporizador con dos salidas, cada una es activada en intervalos distintos, con la funcionalidad de antirrebote en los pines de ingreso y reset.

5.10.1. Descripción detallada

Declaración de la clase timer para manejar dos fases de un temporizador, con antirrebote y un reset.

5.11. timer.h

[Ir a la documentación de este archivo.](#)

```

00001
00006 #ifndef TIMER_H
00007 #define TIMER_H
00008
00009 #include <Arduino.h>
00010
00015 class timer {
00016     public:
00028         timer(int pIngreso, int pReset, int p1, int p2, unsigned long t1, unsigned long t2, unsigned
long rebote);
00029
00034         void iniciar();
00039         void actualizar();
00040
00041     private:
00042         const unsigned long tiempoUno;
00043         const unsigned long tiempoDos;
00044         unsigned long tiempoAntirrebote;
00045         unsigned long ultimoTiempoIngreso;
00046         unsigned long ultimoTiempoReset;
00047         unsigned long cont1;
00048         int pinIngreso;
00049         int pinSalidaUno;
00050         int pinSalidaDos;
00051         int pinReset;
00052         bool contON;
00060         bool antirrebote(int, unsigned long&);
00061 };
00062
00063 #endif
00064
  
```


Índice alfabético

accesoRFID, [7](#)
 accesoRFID, [8](#)

DHT11, [8](#)
 DHT11, [9](#)
 readHumidity, [9](#)
 readTemperature, [9](#)
 readTemperatureHumidity, [10](#)
 setDelay, [10](#)
 setPinDHT11, [10](#)

main/libraries/DHT11-HW-481/DHT11.h, [31](#)
main/libraries/DHT11-HW-481/setPin.cpp, [31](#)
main/libraries/DHT11-HW-481/setPin.h, [33](#), [34](#)
main/libraries/RFID-RC522/accesoRFID.h, [34](#)
main/libraries/RFID-RC522/deprecated.h, [34](#)
main/libraries/RFID-RC522/MFRC522.h, [34](#)
main/libraries/RFID-RC522/MFRC522Extended.h, [39](#)
main/libraries/RFID-RC522/require_cpp11.h, [40](#)
main/libraries/TIMER/timer.h, [40](#), [41](#)
MFRC522, [11](#)
MFRC522::MIFARE_Key, [15](#)
MFRC522::Uid, [15](#)
MFRC522Extended, [16](#)
 PICC_IsNewCardPresent, [27](#)
 PICC_ReadCardSerial, [27](#)
 PICC_Select, [27](#)
MFRC522Extended::Ats, [22](#)
MFRC522Extended::Ats.ta1, [23](#)
MFRC522Extended::Ats.tb1, [23](#)
MFRC522Extended::Ats.tc1, [24](#)
MFRC522Extended::PcbBlock, [24](#)
MFRC522Extended::PcbBlock.inf, [25](#)
MFRC522Extended::PcbBlock.prologue, [25](#)
MFRC522Extended::TagInfo, [26](#)

PICC_IsNewCardPresent
 MFRC522Extended, [27](#)
PICC_ReadCardSerial
 MFRC522Extended, [27](#)
PICC_Select
 MFRC522Extended, [27](#)

readHumidity
 DHT11, [9](#)
readTemperature
 DHT11, [9](#)
readTemperatureHumidity
 DHT11, [10](#)

setDelay
 DHT11, [10](#)
setPin.cpp
 setPinDHT11, [32](#)
setPin.h
 setPinDHT11, [33](#)
setPinDHT11
 DHT11, [10](#)
 setPin.cpp, [32](#)
 setPin.h, [33](#)

timer, [28](#)
 timer, [29](#)