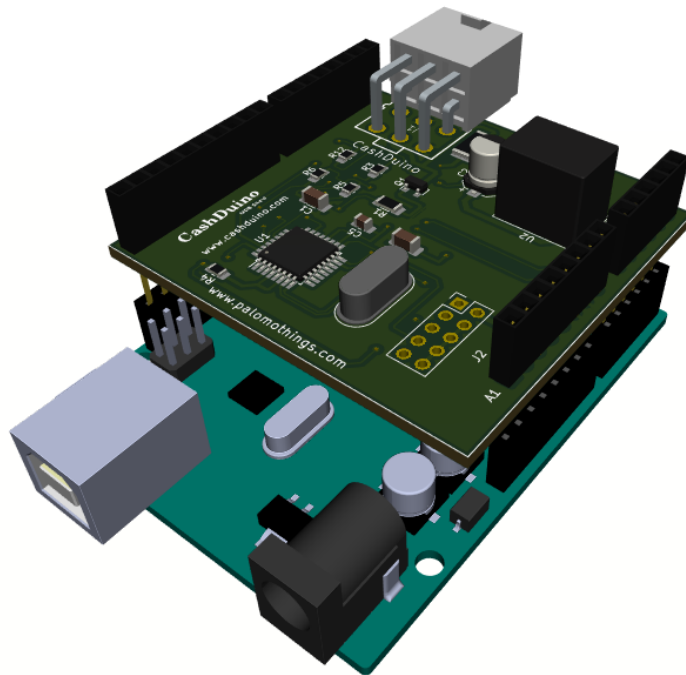


Manual técnico de integración.

CashDuino

MDB Shield



Indice

Control de versión.....	3
Glosario.....	3
Introducción.....	4
Mapeo de pines.....	4
Protocolo.....	5
Tareas.....	5
Eventos.....	7
Código ejemplo.....	9
Github.....	9
Contenido del paquete.....	11



Control de versión

Documento	Manual de integración CashDuino	Fecha: 12/Febrero/2021
Author	Jesus Palomo	Fecha: 12/Febrero/2021
Hardware	Código de producto CashDuino: CDA8-AR Revisión de Hardware: rev01162021AC Código de arnés: CDW-05MDB	Fecha: 10/Noviembre/2020

Glosario

TWI - Two-Wire Interface: El periférico TWI es una interfaz similar al I2C que permite comunicar circuitos integrados.

SDA – Serial Data: Terminal que se utiliza como canal de datos en la comunicación TWI.

SCL – Serial Clock: Terminal que se utiliza como canal de reloj en la comunicación TWI.

I2C – Inter-Integrated Circuit: Protocolo de comunicación serial que permite enviar y recibir información sobre un bus de datos.

MDB – Multi Drop Bus: Protocolo de comunicación single-master, multi-slave que define la comunicación para dispositivos de cobro.

ICP – Internal Communication Protocol – Documento técnico que se usa como referencia de los comandos MDB.

Protocolo: Un sistema con reglas definidas para lograr la comunicación entre dos dispositivos.

Introducción

La tarjeta CashDuino™ permite comunicar y controlar los dispositivos de cobro compatibles con el protocolo MDB y la tarjeta de desarrollo Arduino®. El CashDuino cuenta con puerto TWI (I2C) habilitado para poder comunicarse con tarjetas de control usando una estructura de comandos para ejecutar diferentes tareas.

La arquitectura en alto nivel se muestra a continuación:

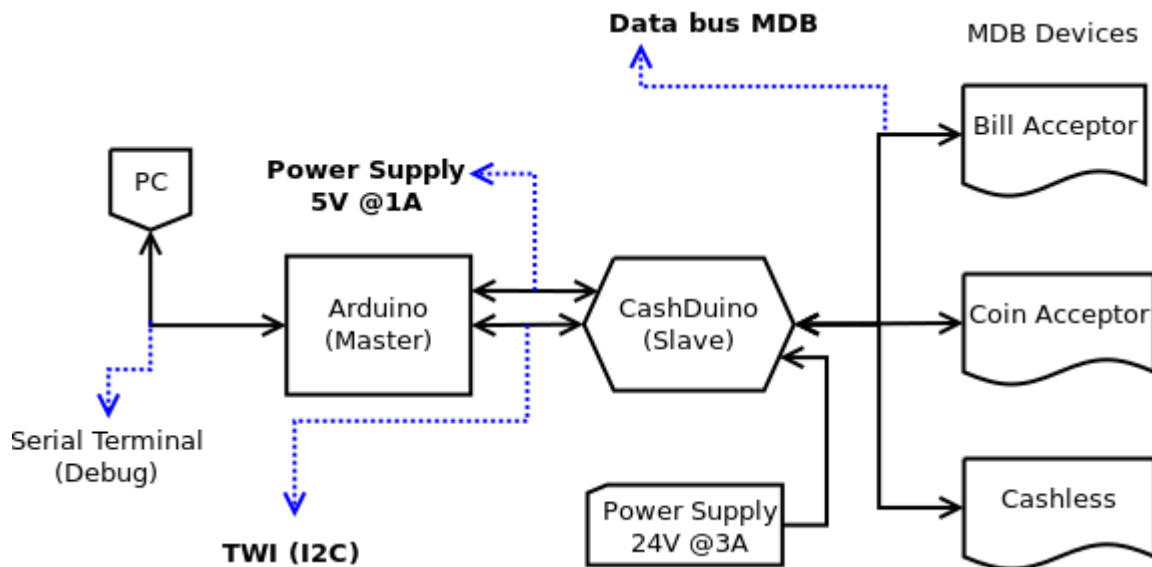
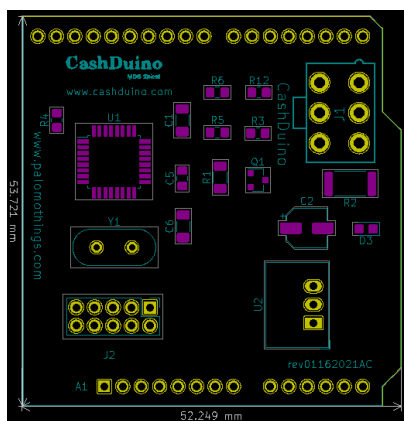


Illustration I: High level Architecture - CashDuino

Maapeo de pines

Arduino	CashDuino
A4 (SDA)	SDA
A5 (SCL)	SCL



Medidas: 52cm x 53cm (Compatible con Arduino R3)

Protocolo

Tareas

Para ejecutar tareas en los dispositivos de cobro la tarjeta Master debe de enviar comandos a través del protocolo TWI, la arquitectura de comunicación se basa en COMANDO/RESPUESTA, por cada tarea (Request) enviada por el Master el CashDuino colocara la respuesta en el buffer de datos.

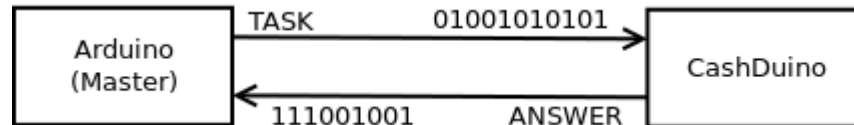


Illustration ii: Master sending a task request

Tabla de comandos para ejecutar tareas, comando enviado por la tarjeta Master y su respuesta.

Tarea	Solicitud Master (Notación Hexadecimal)	Respuesta CashDuino (Notación Hexadecimal)	Descripción.
ENQUIRY BOARD	C1	D1	Comando para solicitar una respuesta del CashDuino.
COIN ENABLE	A1	F1	Comando para habilitar los canales del monedero.
COIN AUDIT	A2	F2	Comando para saber la cantidad de monedas en los tubos del monedero
COIN OUT	A3	F3	Comando para expulsar monedas de los tubos del monedero
BILL ENABLE	B1	E1	Comando para habilitar los canales del billeteiro.
BILL ESCROW	B2	E2	Comando para aceptar o rechazar un billete**

* esta operación es valida solo con billeteiros nivel MDB 3 y con reciclador.

** Esta opción es valida cuando se habilita el ESCROW en billeteiro.

El CashDuino valida una estructura de datos para poder interpretar las tareas solicitadas por la tarjeta Master.

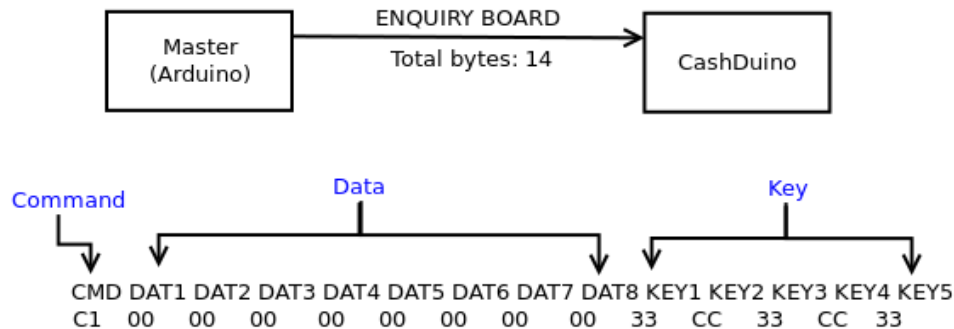


Illustration iii: Data frame format, Master sending ENQUIRY BOARD task (Hexadecimal notation)

Descripción de datos por comando enviado por la tarjeta Master:

COIN ENABLE	DAT1: 0x00 (Coin Enable) DAT2: 0xFF (Coin Enable) DAT3: 0x00 (Manual Dispense Enable) DAT4: 0xFF (Manual Dispense Enable) *Reference: Section 5•9 MDB / ICP
COIN OUT	DAT1 = VALUE (Max 250 Coins) DAT2 = CHANNEL (Coin Type) DAT3 = VALUE (Max 250 Coins) DAT4 = CHANNEL (Coin Type) DAT5 = VALUE (Max 250 Coins) DAT6 = CHANNEL (Coin Type) DAT7 = VALUE (Max 250 Coins) DAT8 = CHANNEL (Coin Type) *Reference: Section 5•10 MDB / ICP
BILL ENABLE	DAT1: 0xFF (Bill Enable) DAT2: 0xFF (Bill Enable) DAT3: 0x00 (Bill Escrow Enable, if this value is 0xFF the ESCROW command is enable) DAT4: 0x00 (Bill Escrow Enable, if this value is 0xFF the ESCROW command is enable) *Reference: Section 6•10 MDB / ICP
BILL ESCROW	DAT1: 0x01 (Escrow status, 0x01 = Insert, 0x00 = Reject) *Reference: Section 6•10 MDB / ICP

Eventos

Los eventos son sucesos que se presentan en los dispositivos de cobro, como insertar un billete, insertar una moneda o presionar la palanca de devolución. El CashDuino coloca la información de los eventos en el bus de datos y es la tarjeta Master quien debe de hacer la petición de datos a través del bus TWI, leyendo los 16 bytes del buffer interno en el CashDuino.

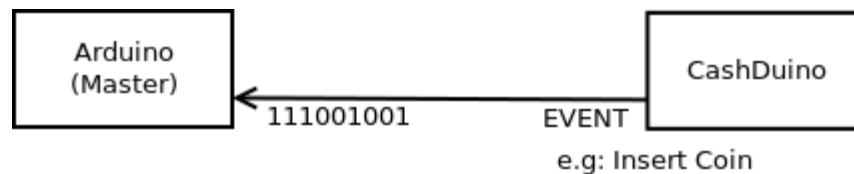


Illustration iv: CashDuino sending an event

Tabla con los comandos que reportan eventos.

Evento	Comando	Descripción
COIN EVENT	F4	Comando que envía el CashDuino cuando detecta un evento en el monedero. Hay 3 eventos reportados en este comando son <i>cash box</i> , <i>tube</i> , y <i>push reject</i> .
BILL EVENT	E5	Comando que envía el CashDuino cuando detecta un evento en el billeteero. Hay 3 eventos reportados en este comando son <i>stacker</i> , <i>pre-stacker</i> , <i>reject</i> .

La tarjeta Master al estar haciendo la lectura del buffer del CashDuino recibirá un paquete de datos con la siguiente estructura.

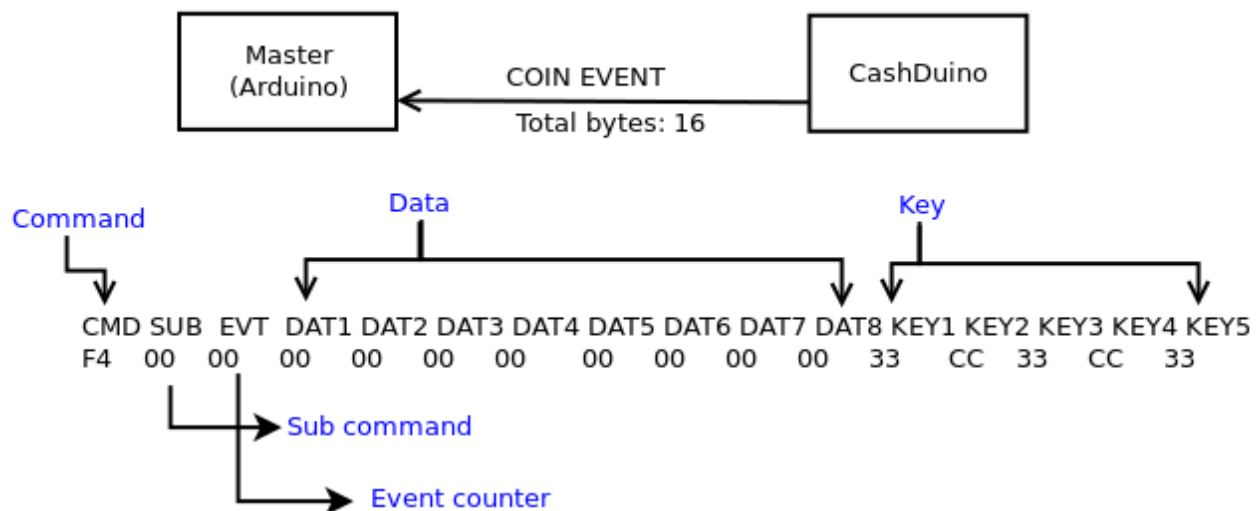


Illustration v: Master reading a tube event from CashDuino (Hexadecimal notation)

Command: valor que indica el tipo de evento.

Sub command: valor que indica la característica del evento, este valor indicara en un valor hexadecimal el código del monto ingresado por canal. Ejemplo: 0x80 indica que se ingreso un billete en el canal 0 (\$20 MXN)

Event counter: El contador de eventos reporta cuando un evento nuevo a sido capturado por el CashDuino.

Data: valor anexos en caso de que la petición del la tarjeta Master requiera una respuesta complementaria.

Key: valores que se usan para validación del paquete de datos.

Descripción de datos por comando:

COIN EVENT	SUB: Coin event (0x40 = Cash box, 0x50 = Tube, 01 = Push Reject) EVT: Event Counter
BILL EVENT	SUB: Bill event (0x80 stacker, 0x90 = pre-stacker, 0xA0 = reject) EVT: Event Counter

La tarjeta Master al solicitar la tarea COIN_AUDIT recibirá la respuesta por parte del CashDuino con la información de la cantidad de monedas por canal almacenado en los tubos del monedero, esta información vendrá por canal desde el byte DAT1 hasta DAT8.

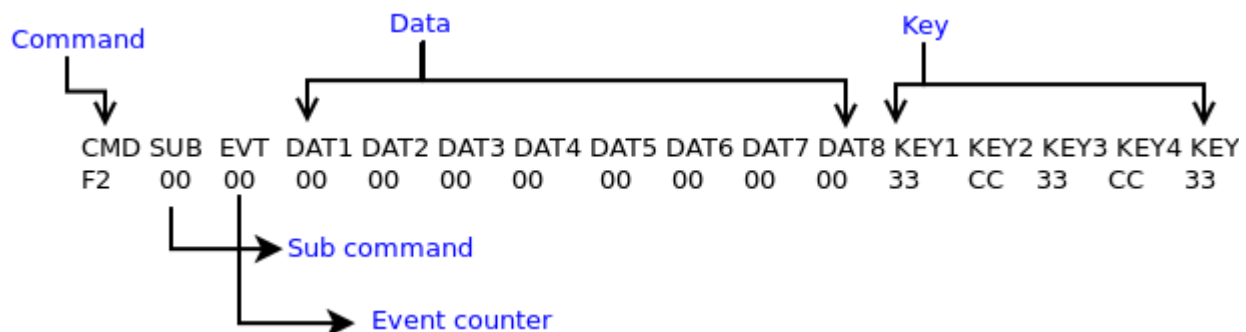


Illustration vi: CashDuino sending tube status response indicating the greatest number of coins that the changer "knows"

Código ejemplo

Es recomendable no tomar este código para producción, su fin es ilustrar el funcionamiento básico del CashDuino así como una guía para el desarrollador al momento de desarrollar su aplicación vending en la tarjeta master.

Importar la librería para manejar el bus TWI y declarar una tarea (comando)

```
#include <Wire.h>
unsigned int CMD_COIN_OUT[] = {0xA3, 0x08, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x33, 0xCC,
0x33, 0xCC, 0x33};
```

Leer el buffer del CashDuino para poder saber el estatus

```
Wire.requestFrom(CASH_DUINO_ADDR, MAX_BUFF_CASHDUINO);
while (Wire.available()) {
  rx_cashduino[cashduino_id] = Wire.read();
  cashduino_id++;
}
```

Detectar cuando un nuevo evento es reportado por el CashDuino

```
if( (command != rx_cashduino[0]) || ( new_event != rx_cashduino[2] ) ){
  command = rx_cashduino[0];
  new_event = rx_cashduino[2];
}
```

Enviar una tarea al CashDuino

```
Wire.beginTransmission(CASH_DUINO_ADDR); // transmit to device CashDuino
for(cashduino_id = 0; cashduino_id<MAX_BUFF_ARDUINO; cashduino_id++){
  Wire.write(CMD_COIN_OUT[cashduino_id]); // sends one byte
}
Wire.endTransmission();
```

Github

En github se encuentra el repositorio que contiene las buenas practicas de código para desarrollar una aplicación vending en la tarjeta master. Existen dos branches, el branch **Master** contiene la versión estable del código de integración mientras que el branch **Develop** se encuentran mejoras y nuevas opciones sujetas a pruebas.

Tips para desarrollar en la tarjeta master:

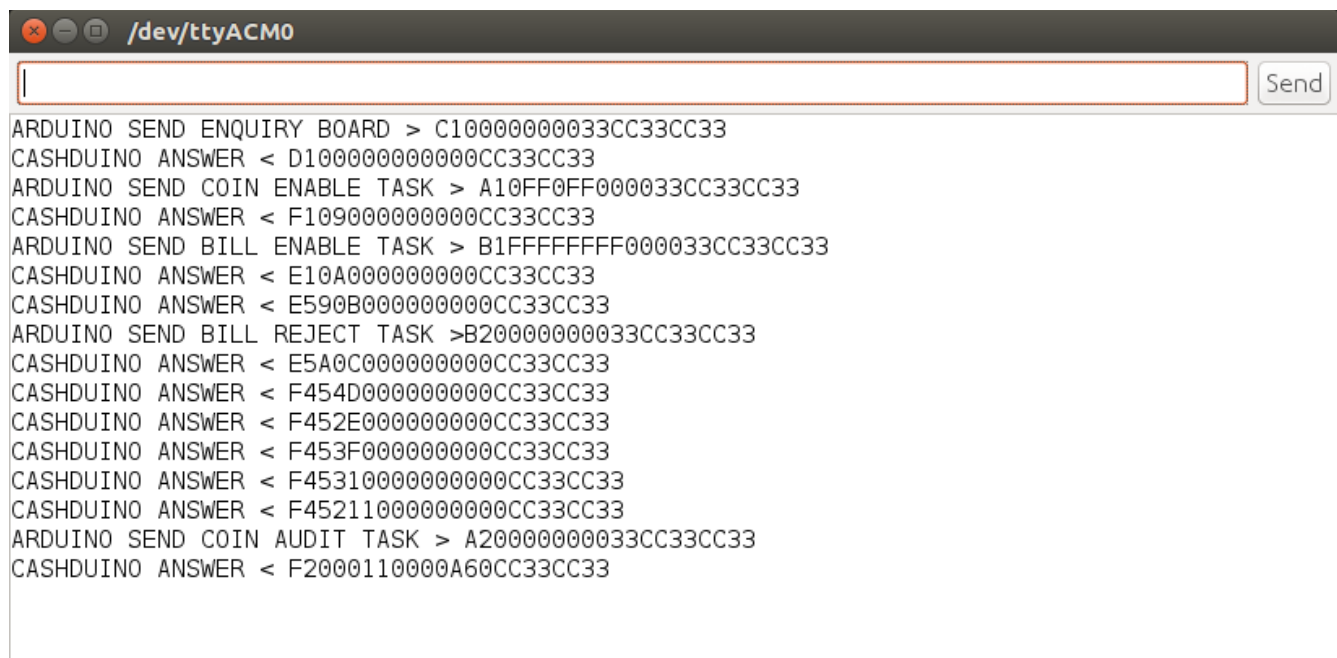
- Cuando la aplicación vending (código ejecutándose en la tarjeta master) necesite atender algún proceso en donde no pueda estar leyendo el buffer del CashDuino, es recomendable des

habilitar los dispositivos de cobro para evitar que el usuario inserte efectivo y no sea contabilizado.

- Se debe de tener un tiempo de espera para que el CashDuino pueda estar procesando el control de los dispositivos de cobro, si se realiza la lectura del buffer del CashDuino de forma muy rápida (<150 ms) se puede ocasionar algún error en el procesamiento del CashDuino, lo recomendable es tener un delay de 200 ~ 250 ms
- Se puede desarrollar la aplicación vending sin necesidad de una computadora, así mismo la tarjeta master puede ser energizada directamente del CashDuino ya que el CashDuino recibe 24V de entrada y regula una salida de 5V.
- Se puede colocar otros Shields encima del CashDuino y así realizar aplicaciones combinando diferentes Shields que den una mayor funcionalidad a la aplicación vending.

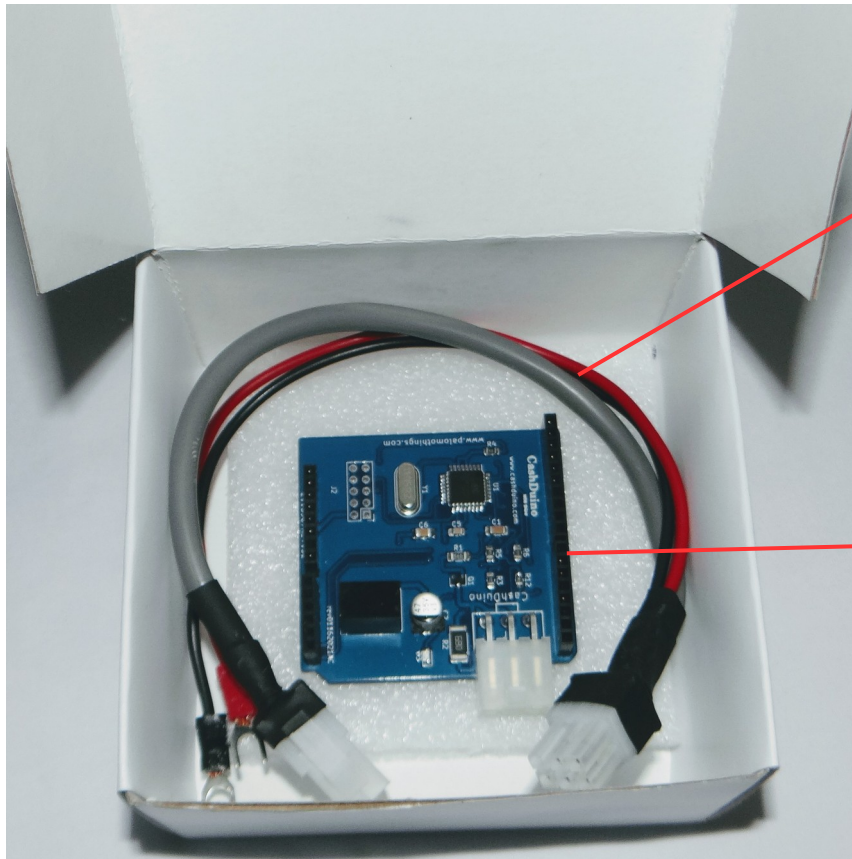
Repositorio: <https://github.com/PalomoProjects/cashduino>

Al ejecutar el programa en el IDE de Arduino y conectar el CashDuino, el código ejemplo `integration_test_cashduino` desplegará la interacción de los comandos usando el Serial Monitor.



```
ARDUINO SEND ENQUIRY BOARD > C10000000033CC33CC33
CASHDUINO ANSWER < D1000000000000CC33CC33
ARDUINO SEND COIN ENABLE TASK > A10FF0FF000033CC33CC33
CASHDUINO ANSWER < F1090000000000CC33CC33
ARDUINO SEND BILL ENABLE TASK > B1FFFFFFF000033CC33CC33
CASHDUINO ANSWER < E10A0000000000CC33CC33
CASHDUINO ANSWER < E590B000000000CC33CC33
ARDUINO SEND BILL REJECT TASK > B20000000033CC33CC33
CASHDUINO ANSWER < E5A0C000000000CC33CC33
CASHDUINO ANSWER < F454D000000000CC33CC33
CASHDUINO ANSWER < F452E000000000CC33CC33
CASHDUINO ANSWER < F453F000000000CC33CC33
CASHDUINO ANSWER < F4531000000000CC33CC33
CASHDUINO ANSWER < F4521100000000CC33CC33
ARDUINO SEND COIN AUDIT TASK > A20000000033CC33CC33
CASHDUINO ANSWER < F2000110000A60CC33CC33
```

Contenido del paquete



Arnes de conexión

Tarjeta electrónica