Bulls-i Terminal

Open Source Data Collection Terminal files - http://hackaday.io/project/1915-data-collection-terminal or http://rodyne.com

Note there are currently 4 parts to the system

1. The PIC32MX1xx/2xx C Firmware Program

this uses the free Microchip MPLABX compiler suite and XC32. Its job is to scan barcodes/rfid's from job-sheets or other barcode/rfid sources such as equipment tags and convert them into small simple wifi message packets (normally under 30 bytes) to be sent to a broker/server program and interpereted/executed.

Note this firmware has custom menus for my customers proprietary MRP factory system which I have mostly left out. I have left the simple job update menu in though which should give you enough info to see how it will work

I have tested the firmware for reading barcodes and RFID MIFAR blocks and reading/writing to the broker program, reading the keyboard and updating the display and these all work so you can basically use the code as a framework to build your own system.

Hardware Schematics here => http://easyeda.com/project_view_Data-Entry-Terminal_neKEWQ8qI.htm

History of project here => http://rodyne.com/?page_id=13

With the current version I am getting about 2 weeks worth of work out of the terminal before needing to recharge which is mainly due to the aggressive turning off of power to modules that are not required and the fact that actual wireless usage in practice is only a few minutes a day.

2. The LUA Code for the ESP01

Rather than using the AT Command set on the first version I have loaded the NodeMCU firmware onto the ESP-01 module (before install!) and created a small script called init.lua which immediately connects to the server on power on the power on/connection takes about 1300mS which means we dont have to deal with sleep modes and power on reset every message, The script is only a few lines long refer to the nodeMCU and lua documentation for further info.

3. The broker program

This is the program which sits in the middle between the terminal and your target system, it must will parse low level requests from the terminal into higher level requests to the target MRP database and vice-versa

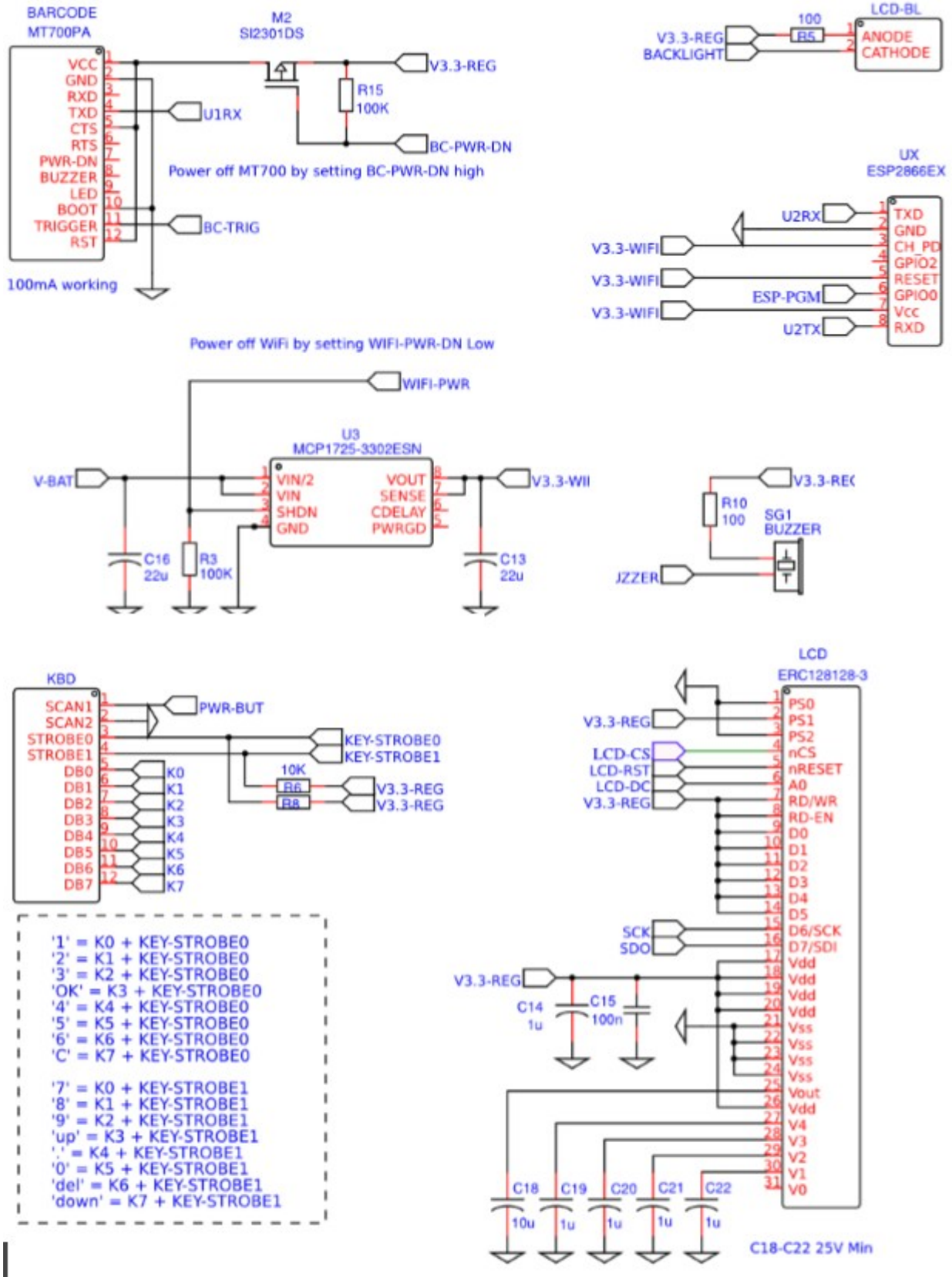
Note with this software I am reading and writing from a customers MySQL database which is proprietary. The program will allow you to connect to your own MySQL database and write your own queries etc but out of the box will not work until you have customised this for your system or wrote your own broker

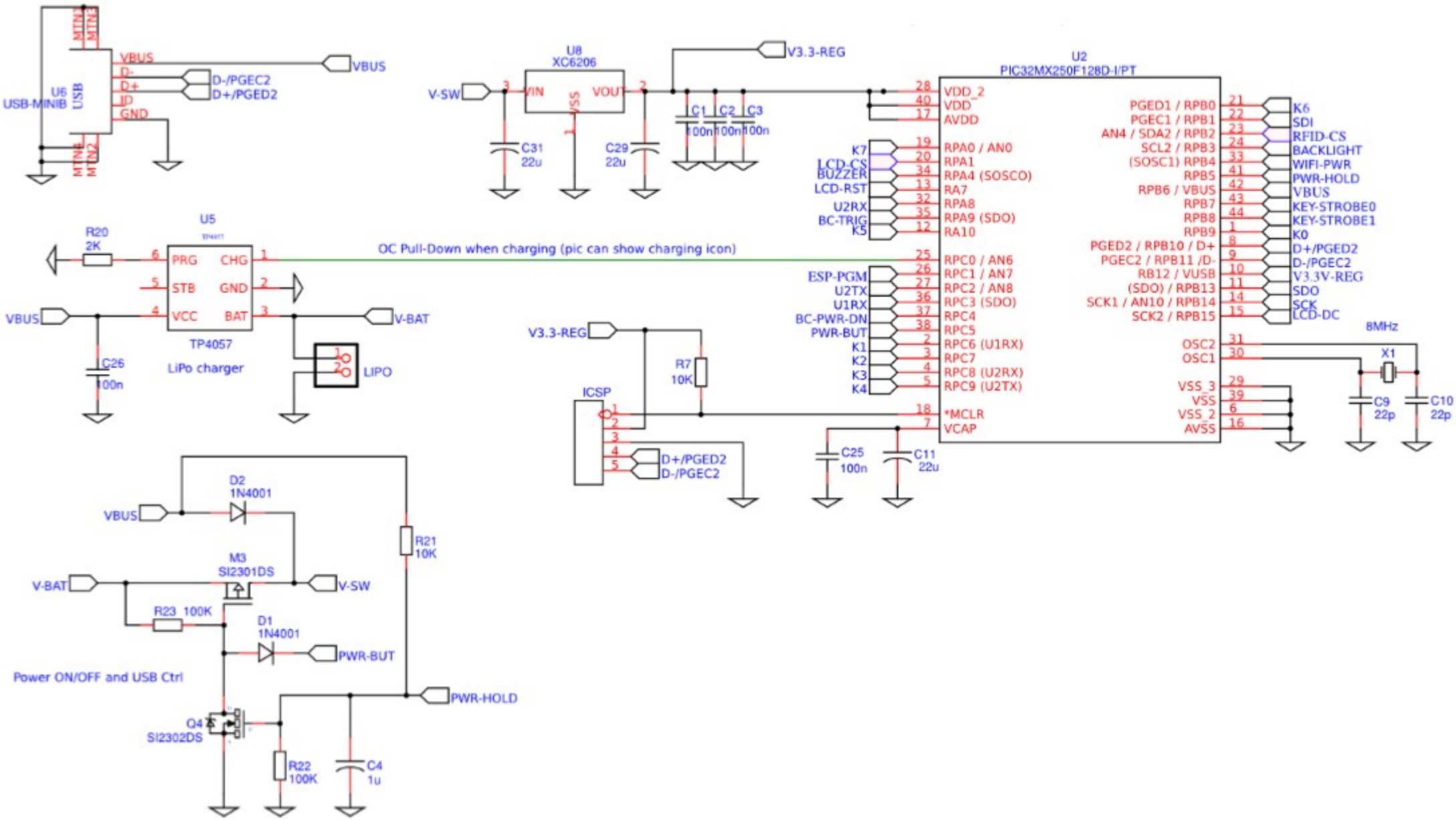
To use this program you will need to add two tables to your database for storage of the message queries and for logging, you will also have to add any extra security as required. I have left an example query and table to show you how a simple message is transferred

4. RFID Programmer

The setup of the wifi is done via programming the SSID, password and broker address into an RFID tag so setting up the device is just a case of going to setup menu and scanning the tag. The tag has 760 bytes of EEPROM I store the SSID in block 1, the password in block 2 and the broker IP address in block 4 (block 3 is reserved for keys etc) the program uses the default mifare key (oxff x 6) so you may want to add an encryption key.

I have also included the code for writing this RFID Tag if you need to write your own


BARCODE
MT700PA
VCC
GND
RXD
TXD
CTS
RTS
PWR-DN
BUZZER
LED
BOOT
TRIGGER
RST
U1RX
BC-TRIG
100mA working
M2
SI2301DS
V3.3-REG
R15
100K
BC-PWR-DN
Power off MT700 by setting BC-PWR-DN high
LCD-BL
100
R5
V3.3-REG
BACKLIGHT
ANODE
CATHODE
UX
ESP2866EX
U2RX
V3.3-WIFI
V3.3-WIFI
V3.3-WIFI
ESP-PGM
U2TX
TXD
GND
CH_PD
GPIO2
RESET
GPIO0
Vcc
RXD
Power off WiFi by setting WIFI-PWR-DN Low
WIFI-PWR
U3
MCP1725-3302ESN
V-BAT
VIN/2
VIN
SHDN
GND
VOUT
SENSE
CDELAY
PWRGD
V3.3-WI
C16
22u
R3
100K
C13
22u
V3.3-REG
R10
100
SG1
BUZZER
JZZER
KBD
SCAN1
SCAN2
STROBE0
STROBE1
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
K0
K1
K2
K3
K4
K5
K6
K7
PWR-BUT
KEY-STROBE0
KEY-STROBE1
10K
R6
R8
V3.3-REG
V3.3-REG
'1' = K0 + KEY-STROBE0
'2' = K1 + KEY-STROBE0
'3' = K2 + KEY-STROBE0
'OK' = K3 + KEY-STROBE0
'4' = K4 + KEY-STROBE0
'5' = K5 + KEY-STROBE0
'6' = K6 + KEY-STROBE0
'C' = K7 + KEY-STROBE0
'7' = K0 + KEY-STROBE1
'8' = K1 + KEY-STROBE1
'9' = K2 + KEY-STROBE1
'up' = K3 + KEY-STROBE1
'.' = K4 + KEY-STROBE1
'0' = K5 + KEY-STROBE1
'del' = K6 + KEY-STROBE1
'down' = K7 + KEY-STROBE1
LCD
ERC128128-3
V3.3-REG
LCD-CS
LCD-RST
LCD-DC
V3.3-REG
SCK
SDO
V3.3-REG
C14
1u
C15
100n
PS0
PS1
PS2
nCS
nRESET
A0
RD/WR
RD-EN
D0
D1
D2
D3
D4
D5
D6/SCK
D7/SDI
Vdd
Vdd
Vdd
Vdd
Vss
Vss
Vss
Vss
Vout
Vdd
V4
V3
V2
V1
V0
C18
10u
C19
1u
C20
1u
C21
1u
C22
1u
C18-C22 25V Min
RFID
U2
RFID-CS
SCK
SDO
SDI
LCD-RST
V3.3-REG
SDA
SCK
MOSI
MISO
IRQ
GND
RST
VDD


U6
USB-MINIB
USB
MTN1
MTN3
MTN4
MTN2
VBUS
D-
D+
ID
GND
D-/PGEC2
D+/PGED2
VBUS
U8
XC6206
V-SW
VIN
VSS
VOUT
C31
22u
C29
22u
C1
C2
C3
100n
100n
100n
V3.3-REG
U2
PIC32MX250F128D-I/PT
VDD_2
VDD
AVDD
RPA0 / AN0
RPA1
RPA4 (SOSCO)
RA7
RPA8
RPA9 (SDO)
RA10
K7
LCD-CS
BUZZER
LCD-RST
U2RX
BC-TRIG
K5
RPC0 / AN6
RPC1 / AN7
RPC2 / AN8
RPC3 (SDO)
RPC4
RPC5
RPC6 (U1RX)
RPC7
RPC8 (U2RX)
RPC9 (U2TX)
ESP-PGM
U2TX
U1RX
BC-PWR-DN
PWR-BUT
K1
K2
K3
K4
*MCLR
VCAP
PGED1 / RPB0
PGEC1 / RPB1
AN4 / SDA2 / RPB2
SCL2 / RPB3
(SOSC1) RPB4
RPB5
RPB6 / VBUS
RPB7
RPB8
RPB9
PGED2 / RPB10 / D+
PGEC2 / RPB11 /D-
RB12 / VUSB
(SDO) / RPB13
SCK1 / AN10 / RPB14
SCK2 / RPB15
K6
SDI
RFID-CS
BACKLIGHT
WIFI-PWR
PWR-HOLD
VBUS
KEY-STROBE0
KEY-STROBE1
K0
D+/PGED2
D-/PGEC2
V3.3V-REG
SDO
SCK
LCD-DC
OSC2
OSC1
VSS_3
VSS
VSS_2
AVSS
8MHz
X1
C9
22p
C10
22p
C25
100n
C11
22u
R7
10K
ICSP
D+/PGED2
D-/PGEC2
U5
R20
2K
PRG
CHG
STB
GND
VCC
BAT
TP4057
LiPo charger
VBUS
C26
100n
V-BAT
LIPO
OC Pull-Down when charging (pic can show charging icon)
D2
1N4001
VBUS
M3
SI2301DS
V-BAT
V-SW
R23 100K
D1
1N4001
PWR-BUT
Power ON/OFF and USB Ctrl
R21
10K
PWR-HOLD
Q4
SI2302DS
R22
100K
C4
1u

Wifi Module configuration

The ESP-01 is loaded with Node MCU V0.96 firmware and the following script is loaded and saved as init.lua so it starts automatically

```
sk = net.createConnection(net.TCP, 0)

sk:on("receive",  function(sk, c) print(c) end )
sk:on("connection",  function() print("Connected") end )

tmr.alarm(1,1000, 1,
  function()
    if wifi.sta.getip()~=nil then
      tmr.stop(1)
      print("\r\nOK " .. wifi.sta.getip() .. " in " .. tmr.now()/1000 .. " mS")
      if file.list()["host.cfg"] then
        file.open("host.cfg", "r")
          server = file.readline():gsub("%s+$", "")
        file.close()
        sk:connect(2001,server)
        end
    elseif tmr.time()>10 then
      tmr.stop(1)
      print("\r\nFail!")
    end
  end
)

-- both SSID and Password are CaSe SeNsITiVe! and must be enclosed in quotes!
-- eg SetWIFI("Indevin","Password123","192.168.0.2")

function SetWIFI(SSID,pwd,host)
  wifi.setmode(wifi.STATION)
  wifi.sta.config(SSID,pwd)
  wifi.sta.autoconnect(1)
  file.remove("host.cfg")
  file.open("host.cfg", "w+")
  file.writeline(host)
  file.close()
  print("\r\nOK")
end
```

On boot ithe timer calls every second and attempts to automatically connects to last wifi and the host which was saved

When the terminial receives the “OK” message it should send the data to the broker using

eg

sk:send("mydata")

The callback func should print whats received on serial port directly to the serial

If no connection the terminal can use the SetWIFI function to save the configuration

eg

SetWIFI(“CYBERNET”,”MadeUpSSIDPassword”,”192.168.0.2”)

BOTH SSID and PASSWORD are case sensitive!

Note the WiFi module is normally powered off and open powered on just before send, it therefore does a full POR each time we use it. After message is received from broker it is powered off again immediately to preserve battery. The startup current for the ESP8266 is about 300-400mA (for a few milliseconds) then drops to 70mA and about 200mA for tx/rx if the batteries are a bit low we may start to see more failures as an indication the battery needs charging. The high startup current is also the reason it has its own LDO.

Bulls-i Mobile Data Message Format

WiFi link must be encrypted AES2 minimum. Text Encoding Standard ASCII

Data Format

TAB-DELIMITED-DATA	CR (0x0D)	LF (0x0A)

Note All queries require the mobile device to be assigned to a user, ithe system will do this if the UserPWMD5Hash field is set and matches a valid account. f not assigned then retuns unnassigned error. Actual SQL Queries are stored in query table using function to lookup SQL and fields substituted real-time to form query. All messages are initiated from the mobile terminal.

GET-JOBSTEP

Mobile Device => Broker (Initiator)

Field0 20
Field1 Device UID (NodeMCU GetCPUID) – Unique to each wifi chip can use to id/register terminal
Field2 UserPWMD5hash (or blank)
Field3 Barcode as scanned including prefix eg “X123456” (for JIID)
Field4 <blank>

Broker => Mobile Device (Responder)

Field0 0=OK or 1 = Not Authorised or 2=Job Not Issued
Field1 Job Number
Field2 Metric Short Code (text to display eg “Enter final DIP”)
Field3 Metric Type:
0 = RFID Scan
1 = Barcode Scan
2 = Integer
3 = Decimal
4 = Time or duration (HH.MM)
Field4 UOM Text (“”, “cm”, ”Ltr”, “Kg”, “DegC”
Field5 Metric Min when Field4 = 2 or 3 (number)
Field6 Metric Max when Field4 = 2 or 3 (number)
Field7 <Blank>
Field8 <Blank>

SET-JOBSTEP

Mobile Device => Broker (Initiator)

Field0 21
Field1 Device UID (as above)
Field2 UserPWMD5Hash (compulsory)
Field3 Barcode as scanned including prefix letter eg “X123456” (for JIID)
Field4 Metric as text string (if data capture is required) eg “2355”

Broker => Mobile Device (Responder)

Field0 0=OK or 1 = Not Authorised or 2=Job Not Issued
Field1 <Blank>
Field2 <Blank>
Field3 <Blank>
Field4 <Blank>
Field5 <Blank>
Field6 <Blank>
Field7 <Blank>
Field8 <Blank>

GET-ANALYSIS

Mobile Device => Broker (Initiator)

Field1 30
Field2 Device UID (NodeMCU GetCPUID) – Unique to each wifi chip can use to id/register terminal
Field3 UserPWMD5hash (or blank)
Field4 Barcode as scanned including prefix eg “T123456” (for TestID)
Field5 <blank>

Broker => Mobile Device (Responder)

Field1 0=OK or 1 = Not Authorised or 2=Job Not Issued
Field2 Analysis Number (LabID)
Field3 Test ShortCode (text to display eg “pH”)
Field4 Metric Type:
0 = Pass/Fail
1 = Barcode Scan
2 = Integer
3 = Decimal
4 =
Field5 UOM Text (“”, “ppm”, “DegC” etc)
Field6 Metric Min when Field4 = 2 or 3 (number)
Field7 Metric Max when Field4 = 2 or 3 (number)
Field8 <Blank>
Field9 <Blank>

SET-ANALYSIS

Mobile Device => Broker (Initiator)

Field1 31
Field2 Device UID (as above)
Field3 UserPWMD5Hash (compulsory)
Field4 Barcode as scanned including prefix letter eg “T123456” (for TestID)
Field5 Metric as text string eg “2.355”

Broker => Mobile Device (Responder)

Field1 0=OK or 1 = Not Authorised or 2=Job Not Issued
Field2 <Blank>
Field3 <Blank>
Field4 <Blank>
Field5 <Blank>
Field6 <Blank>
Field7 <Blank>
Field8 <Blank>
Field9 <Blank>