Senseceiver

Product Manual

Rev .04 – 17-02-2017

Contents

Product Description

The Round Solutions Senseceiver is an embedded IoT/M2M hardware that simplifies Internet Connectivity to Internet or Cloud. The hardware is built around the Telit Cellular module Series XE910. The device comes in different module variants supporting 2G, 3G, and 4G. GPS functionality is also available as an optional feature. It can be used to easily make physical devices connect to the web. It's literally plug, push and play.
The device is available in different variants: Senseceiver UART, Senseceiver I/O, and Senseceiver 4-20mA.

1- Device Variants

a. Senseceiver UART: Provides TTL serial connection to Telit module. Logic level voltage is 3.3V. Digital input 1 can be used to reset the module in case needed by holding it high for at least 5 seconds. Releasing it turns the module back on.
b. Senseceiver I/O: Provides out of the box connectivity to Cloud by sending the following measurements to it: Digital Inputs and Analog inputs. Digital outputs can be used to turn sensors on to read data from. The device can be scheduled to wake up at certain intervals to read data from sensors and send it to the cloud. *
c. Senseceiver 4-20: Provides out of the box connectivity to Cloud by sending the following measurements to it: 2 x Current measurement loop (4-20mA). Digital outputs can be used to turn sensors on to read data from. The device can be scheduled to wake up at certain intervals to read data from sensors and send it to the cloud. *

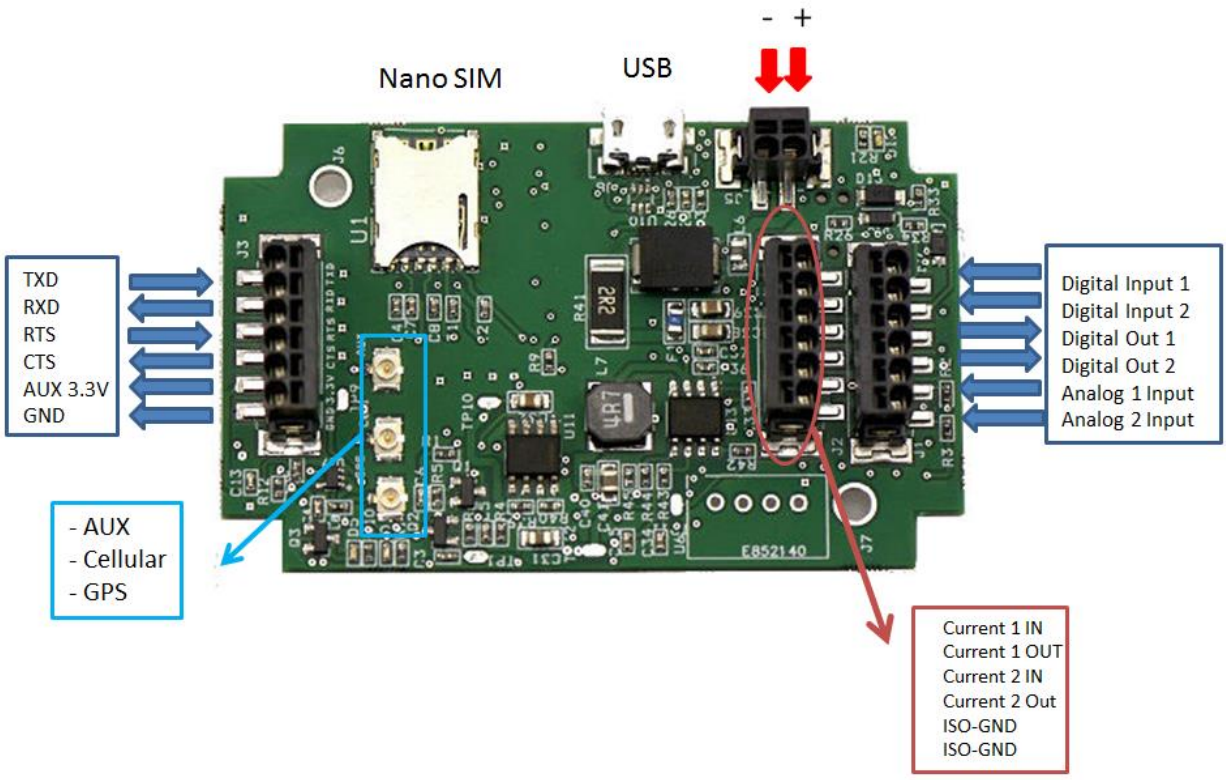
*Source code is available on request.

2- Key Benefits

- Easy to integrate with other devices & sensors using USB, UART, High Precision Analog Input, High Precision 4-20mA current inputs, and Digital Inputs.
- Supports 2 Digital open collector outputs.
- Battery Friendly. Supports sleep mode where the module can be schedules to wake up at certain intervals.
- Increased connectivity performance with 2G/3G/4G
- Ideal platform for m2m applications and mobile data
- Programmable via Python and Telit's App Zone C
- Combines high-speed wireless connectivity combined with embedded multi-constellation high-sensitivity positioning GPS + GLONASS receiver
- Internet friendly with integrated TCP/IP and UDP/IP stacks
- Simple drop-in migration and technology design reuse path to 2G, 3G and 4G² with any Telit xE910 cellular module
- Nano SIM card holder – SIM On Chip Footprint available for 1000+ Quantity orders
- Wide range power supply input 5-28 V DC

3- Interfaces

- 2 x Analog Inputs (0-10VDC)
- 2 x 4-20mA Current measurement loop (Isolation is available as option)
- USB interface
- UART Interface
- 2 x Digital Inputs (0-28VDC)
- 2 x Open Collector outputs
- Antenna Connectors
    - GSM: U.FL Connector
    - Cellular Aux :UFL Connector
    - GNSS: U.FL Connector



4- Development Environment:

The Senseceiver devices come as Out Of The Box ready devices but also ready for programming in Python of Telit App Zone C. Please consult with Round Solutions on your project and we will advise how to move forward. The client can develop the apps or can commission Round Solutions to do it.

5- USB Driver:

The Telit module on the Senseceiver creates Virtual COM Ports (VCP) that the user can use to communicate with the module. The VCP driver can be provided to customers of Round Solutions. Two VCPs are connected to an AT parser and one of these two can send and receive data from the script.

6- RSTerm – Terminal Software:

We recommend using the RSTerm terminal software available on our website to download scripts to the Senseceiver. The RSterm can be downloaded from:

http://www.roundsolutions.com/media/pdf/PCB-NT-GE866_rsterm.zip

Select **Python** view from the Top menu and use the below buttons to list, write, read, select, and set as main Python scripts to and on the Senseceiver.

AT#LSCRIPT: Lists the scripts inside the module

AT#WSCRIPT: Opens a file browser and let you choose the file you want to write to the Senseceiver

AT#ESCRIPT=”xxx.pyc” : set the selected script as the main executing script

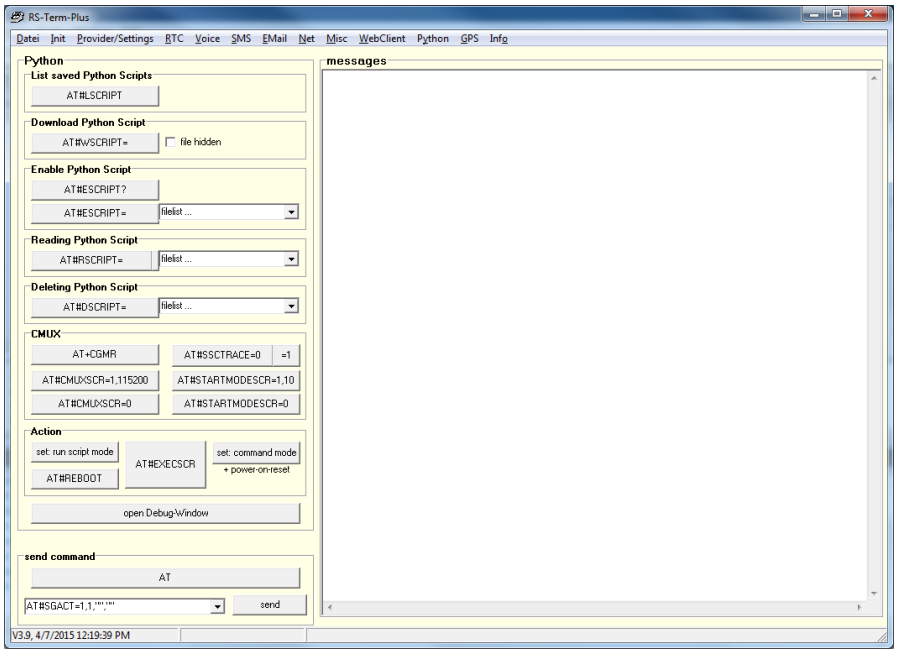
AT#DSCRIPT=”xxx.pyc” : deleted the selected script

AT#EXECSCR: Executes the script selected as the main executing script

AT#STARTMODESCR=1,10 : Executes the script after each boot of the module within 10 seconds if no AT commands are sent.

AT#STARTMODESCR=0: Disables the previously mentioned feature.

Here’s how the **Python** view looks like



7- Python Class:

The Senseceiver is provided with a Python Class to facilitate developing applications. The class provides encapsulation of AT commands sending and handling. The file MODEM_RS.pyc should be downloaded on the module and imported in the main script.

Modem_RS:

Here's a code snippet that shows how the Modem_RS class works:

```
import MODEM_RS

m_modemmanager = MODEM_RS. Modem_RS ()

m_modemmanager.ExecuteATCommand("Sample AT Command",TimeOut)  #executes the commands and returns the response

m_modemmanager.SetAPN("APN Of Operator") # returns 1 is successful and 0 if not

ret = m_modemmanager.ActivateContext("Username","password")  #returns 1 if context if activated, 0 otherwise

m_modemmanager.DeactivateContext()

ret = m_modemmanager.IsContextActive()  #returns 1 of context is active, 0 otherwise

ret  = m_modemmanager.GetDCD()  #returns 1 if DCD is high

ret = m_modemmanager.ConnectToServer("Server_Address or IP",TCP_Port)

m_modemmanager.SendData("sample data to send when connection to the server is established")

m_modemmanager.DisconnectServer()  #closes the connection to the server

m_modemmanager.InitSMSSettings()  #call once at the beginning of the script

m_modemmanager.SendSMS("Number","SMS Contents")
```

8- Reading IO Data:

The Analog inputs, Current Inputs, digital Inputs and outputs are measured and controlled by the microcontroller.

The Telit module can request the inputs measurements from the microcontroller and can also request from the microcontroller to turn it off for a certain period. Please use the function GetPicInputs() to read the values of the IOs. The variables Analog1, Analog2, CIN1, CIN2, DIN1, DIN2, MainVoltage, DOUT1, and DOUT2 are defined globally as demonstrated by the example script. Please refer to the sample file testcom.py for an example on how to use this function.

```
def GetPicInputs():
    SER2.send('\xAA')
    time.sleep(2)
    PICanswer = SER2.read()
    global Analog1
    global Analog2
    global CIN1
    global CIN2
    global DIN1
    global DIN2
    global MainVoltage
    global DOUT1
    global DOUT2
    if(len(PICanswer)>14):
        Analog1 = ord(PICanswer[1:2])
        Analog1 = Analog1 * 256
        Analog1 = Analog1 + ord(PICanswer[2:3])

        Analog1 = Analog1 * 625
        Analog1 = Analog1 *122
        Analog1 = Analog1 / 22
        Analog1 = Analog1 /10000


        Analog2 = ord(PICanswer[3:4])
        Analog2 = Analog2 * 256
        Analog2 = Analog2 + ord(PICanswer[4:5])
        Analog2 = Analog2 * 625
        Analog2 = Analog2 *122
        Analog2 = Analog2 / 22
        Analog2 = Analog2 /10000

        CIN1 = ord(PICanswer[5:6])
        CIN1 = CIN1 * 256
        CIN1 = CIN1 + ord(PICanswer[6:7])
        if(CIN1>32767):
            CIN1 = 65536 - CIN1
        CIN1 = CIN1 * 625
        CIN1 = CIN1 /10000
        CIN1 = CIN1 / 49.9
        CIN1 = int(CIN1*100)
        CIN1 = CIN1/100.0


        CIN2 = ord(PICanswer[7:8])
        CIN2 = CIN2 * 256
        CIN2 = CIN2 + ord(PICanswer[8:9])
        if(CIN2>32767):
            CIN2 = 65536 - CIN2
        CIN2 = CIN2 * 625
```

```
CIN2 = CIN2 /10000
CIN2 = CIN2 /49.9
CIN2 = int(CIN2*100)
CIN2 = CIN2/100.0

MainVoltage = ord(PICanswer[9:10])
MainVoltage = MainVoltage * 256
MainVoltage = MainVoltage + ord(PICanswer[10:11])
MainVoltage = MainVoltage * 1000
MainVoltage = MainVoltage * 19
MainVoltage = MainVoltage * 18
MainVoltage = MainVoltage /1023
MainVoltage = MainVoltage / 10

DIN1 = PICanswer[11:12]
DIN2 = PICanswer[12:13]
DOUT1 = PICanswer[13:14]
DOUT2 = PICanswer[14:15]
```

To shut down the Telit module and schedule a wakeup, please use the following code:

```
SleepTime = 10 #in minutes
HighByte = int(SleepTime / 256)
LowByte = SleepTime - (HighByte * 256)
SleepStr = ' \xBBab'
SleepStr = SleepStr.replace('a',chr(HighByte))
SleepStr = SleepStr.replace('b',chr(LowByte))
SER2.send(SleepStr)
time.sleep(1)
```

To change a variable value by sending a SMS to the device, the following code could be used:

```
#assume that you send a SMS to the device with the following text:  interval:30,
#the comma at the end is necessary to determine the end of the message. Any other character can also be used
def CheckSMS():
   SMSBuffer = m_modemmanager.ExecuteATCommand('AT+CMGL=ALL\r', 20)
   headindex = SMSBuffer.lower().find('interval:')
   if(headindex!=-1):
      tailindex = SMSBuffer.find(',',headindex)
      NewInterval = SMSBuffer[headindex+9:tailindex]
      m_modemmanager.ExecuteATCommand('AT+CMGD=1,4\r', 5)   #deletes the SMS in the modukle
      m_modemmanager.SendSMS('+49123456789',"Interval is updated ") #send back a confirmation SMS
```

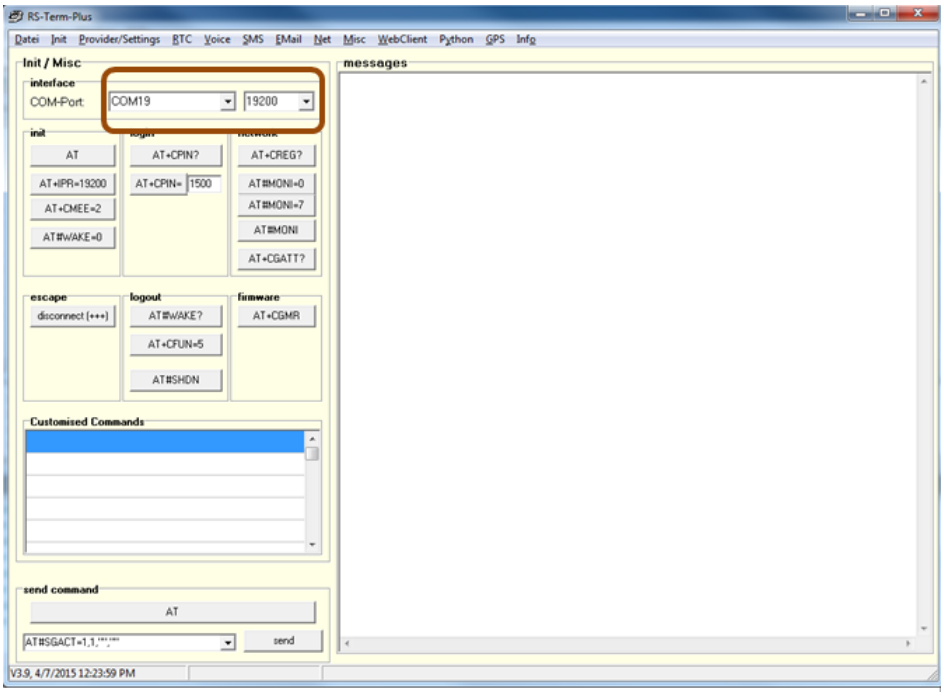
Note that it is necessary to save the new value of the variable in a file and read it at next startup. This is done through standard code in Python for file Read/Write.

9- Device Main Script:

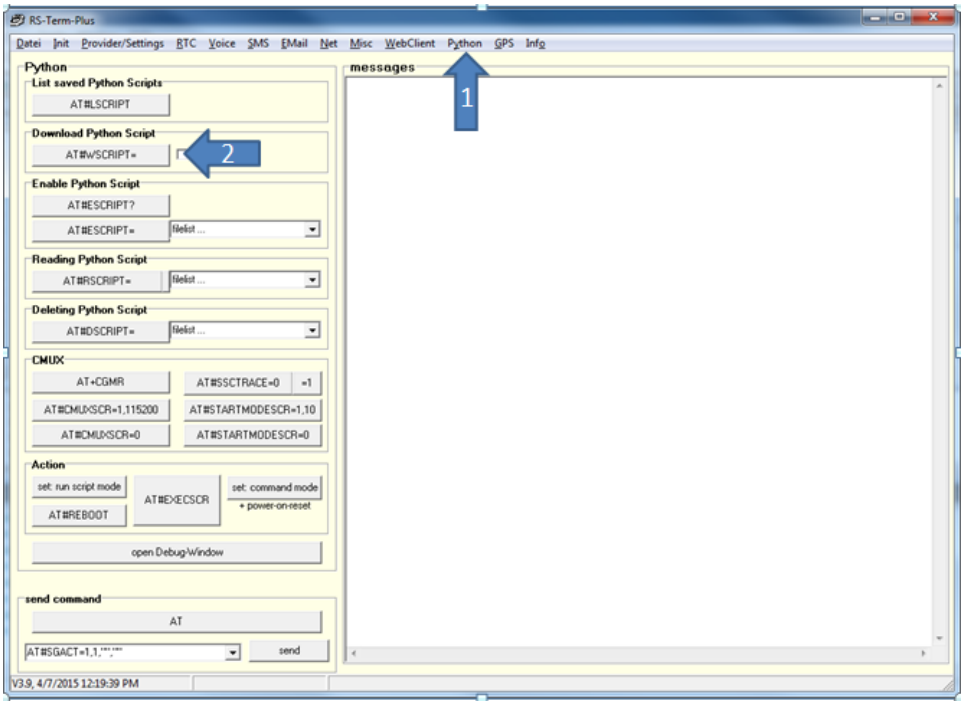
Unless otherwise mentioned, the device doesn't come with a script preloaded. Please contact Round Solutions to get example source code to download to the device.

Before proceeding, make sure you have successfully installed the USB drivers.

Upon opening RSTerm, make sure you select the proper COM port and set the baud rate as shown below. Click on the button AT and wait for the tracker to reply by OK to make sure you have established communications with the module.



After that click on **Python** on the Top menu, then click AT#WSCRIPT. Note that you cannot overwrite a script with the same name that already exists. You need to delete it first from the module and then download the new one.



Steps to download a new Python script:

Downloading a new script using RSTerm terminal software is very easy. Just follow the following steps after you have successfully connected the device:

1- Go to Python View by clicking on **Python** on the top men
2- Click on **AT#LSCRIPT** to list the scripts downloaded to the device
3- If the script name already exists, make you sure you delete it using **AT#DSCRIPT=”scriptname.pyc”**
4- Click on **AT#WSCRIPT** and navigate to the script file you wish to download
5- Once finished, click on **AT#LSCRIPT** again to update the populated lists with the name of the new script
6- Type in the script name in the field next to the button **AT#ESCRIPT=** and then click on the button
7- Click on the button **AT#STARTMODESCR=1,10** to enable running the script upon module boot after 10 seconds
8- Click on **AT#EXECSCR** to execute the script immediately

Connect a TCP socket Example Code:

```
import USB0

import MODEM_RS

import MDM

import MOD

import time

GPRS_USER = ""

GPRS_PASSW = ""

M2MAPN = "surfo2"

m_modemmanager = MODEM_RS.Modem_RS()

m_modemmanager.SetAPN(APN)

if(m_modemmanager.ActivateContext(GPRS_USER,GPRS_PASSW)==1):

        if(m_modemmanager.ConnectToServer("Server_Address",80)==1):

                m_modemmanager.SendData("Some data to server")
```