SerialOTG **Ver 1.2**

Tested with the following serial adapters:
PL2303, PL2303HX, (PL2303HXN not implemented yet)
FTDI FT232
CP210x
CH341, CH341 (“fake”)
CDC-ACM: Original Arduino Uno, Adafruit Feather_M0 and CLUE, Micro:bit, Teensy, OpenCR
CDC-ACM: ATTINY85 using DigiCDC library
BT tested with BT-terminal, HC-05, upload with HC-05/UNO
WIFI tested with wifi tcp serial terminal and tcp server app.s

Set up methods

Initialize()	Initialize for USB OTG serial communication
InitializeBT(Btname)	Initialize for previous paired BT client communication if Btname=”” Initialize as server (For 1 client)
InitializeWIFI(IP-address, port)	Initialize for TCP WIFI client communication. If IP-address=”” Initialize as server (For 1 client)

Open()
Open a serial connection for the adapter. Return true if opened.
Sets default values: baud=9600, parity=none, Rts=0, Dtr=0, (databits=8, Stopbits=1)
For BT client Try to connect. Return true if success.
For BT server. Start server task that waits for a connection. Return true if started.
For WIFI client Try to connect. Return true if success.
For WIFI server. Start server task that waits for a connection. Return true if started.

Close()

Properties

IsInitialized()	Return true if initialized
IsOpen()	Return true if opened
IsConnected()	Return true if connected (for USB serial return same as IsOpen())

BaudRate(int baudRate) Set baudrate. Return true or false.
BaudRate(). Return baudrate.
Baudrate (300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200)

Parity(int parity). Set parity. Return true or false.
Parity(). Return parity.
Parity 0=NONE, 1=ODD, 2=EVEN, (3=MARK 4=SPACE)

Dtr(int dtr). Set Dtr. Return true or false.
Dtr(). Return Dtr.
Dtr 0, 1

Rts(int rts). Set Rts. Return true or false.
Rts(). Return Rts.
Rts 0, 1

DriverName()	Return driver name. Identifies the USB-Serial adapter. Current results are: -.UartCdcAcm -.UartCp210x -.UartFtdi -.UartPL2303 -.UartWinCH34x -.UartWIFI -.UartBlueTooth

Communication methods
The extension gives you different possibilities to communicate over a serial line. All read methods returns the characters received up to the moment it is called, except for ReadLn() that collects characters at each call until an eol character is found.

Using UTF-8 string
Read()
Returns UTF-8 string, empty string if nothing to read.

Write(String data)
Write UTF-8 string, return true or false.

Using Hex coded Ascii
ReadHex()
Read bytes from serial line. Return a hex-coded string with each byte represented by 2 characters (0..9 A..F). If no data, return empty string.

WriteHex(string)
Write a hex-coded string as bytes to serial line.
Uses 2 hex digits for each byte. Uses digit 0..9 and A..F so no problem with UTF-8 coding

Using Ascii message
ReadLn()
Read serial line and collect characters up to new_line. If new_line found, return string, else return empty string.

WriteLn(String data)
Write string, append new_line. Return true or false.

Using single byte
ReadByte()
Read a byte as a number 0..255 from serial line, if empty return -1

WriteByte(num)
Write a number 0..255 as a byte to serial line

Using multiple bytes in lists
ReadBytes()
Read unsigned bytes 0..255 from serial line to a list.

WriteBytes(List)
Write unsigned bytes 0..255 in a list, to serial line.

Buffer control
Available() Returns current nr of bytes in driver read buffer. 0 if empty.

Flush() Empty the internal read buffers.

Arduino uppload
This function can be used for upload of Arduino hex files over serial OTG, Bluetooth (and wifi).

Int Upload(board,filename) Upload hex file to Arduino. Return 0 if ok, else error nr.
Boards:
1=UNO, 328 115200 STK500
2=MEGA2560, 2560 115200 STK500V2
3=Nano328, 328 57600 STK500
4=Mini328, 328 57600 STK500
5=Pro/Pro mini 5V 328, 57600 STK500
6=Pro/Pro Mini 3.3V 328 57600 STK500
7=Pro/Pro Mini 5v/3.3V168 19200 STK500

int UploadProgress() Return block nr sent.

CancelUpload() Cancel the upload.

Some notes on SerialOTG

Initialize must be called first, before OpenSerial()

First call to Open() detects the adapter and you must allow USB access. Second call opens the serial connection.

Some adapters requires RTS=1 (and/or DTR=1) to communicate. This is true for Arduino Pro Micro, 32U4 chip implementing CdcAcm.

Dtr is used to reset Arduino.

The library functions is executed in the same thread as the calling program. This means that the AI2 UI can be blocked during large transfers or if you uses a wait loop. Use a timer instead of an wait loop.

Internal read and write buffer size is 1024 Bytes. Internal read message buffer for ReadLn() is 256 Bytes.

Read and Write communication methods are independent. You can mix them as you want.

Remember that serial line communication is just a stream of bytes. There is no packet handling. Each byte takes some time to transfer on the line. So if you send a request, then you have to wait for a while, before you get an answer.
You have to add some kind of message handling yourself by timing, message length, content etc.

WriteLn(), ReadLn() implements a kind of message handling. It uses new line character to end an UTF-8 message. Use ASCII char set (7-bit) to avoid problems if you communicate with Arduino.

Close your serial connection before you make any changes to your app or refresh the screen.

If you need to reset the connection from the PC, remove the current instance of the Ai2 Companion in the phone. Do not just restart with the companion icon.