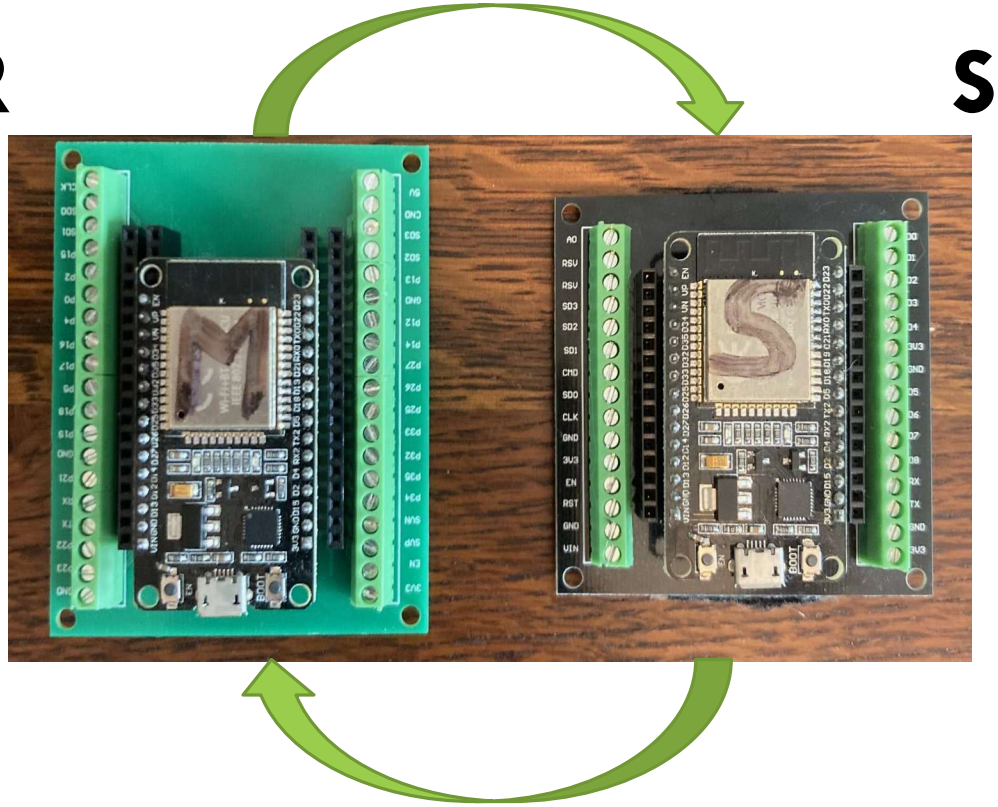
ESP32Forth Bluetooth

Project: 2 ESP32 DevMods running ESP32Forth talking to each other.

ESP32Forth 7.0.7.10 **Arduino 1.8.19 & 2.0.4**

MASTER **SLAVE**



ESP32Forth Bluetooth

NOTES

The Arduino vs 1.8.19 & 2.0.4 will run simultaneously. This way you can have both the Master and Slave units connected.

Only the Master unit can initiate a connection to a Slave unit.

The Slave unit has to be up and running **FIRST**, making it visible to the connecting Master unit.

If you have the 'Core Debug Level' in Arduino tools set to '**NONE**' you will not see any error messages, should you try any other connection.

Recommend setting the level to **'Verbose'** during your initial trials.

ESP32 DevMod Bluetooth is **NOT** Enabled

ESP32Forth.ino looks for PSRAM and if not available or

Disabled then Forth does not enable the BT.

FORCE the ENABLE : Add to the ESP32Forth.ino the following:

#define ENABLE_SERIAL_BLUETOOTH_SUPPORT

in section

// Default on several options (vs 7.0.7.10 line 67-79)

ESP32forth v7.0.7.10 - rev c43cbfd9d91ed1b2511b
ESP32-D0WDQ6 240 MHz 2 cores 4194304 bytes flash
System Heap: 95396 free + 235944 used = 331340
total (28% free) 65524 bytes max contiguous
Forth dictionary: 70232 free + 34100 used = 104332 total
(67% free)
3 x Forth stacks: 2048 bytes each

Needs to be > 120000

also internals also esp
150000 getfreeheap – relinquish

(Add above to an autoexec.fs file in SPIFFS)

Memory Hog:

Sketch uses 1709657 bytes (**81%**) of program storage space.

Maximum is 2097152 bytes.

Global variables use 62504 bytes (19%) of dynamic memory, leaving 265176 bytes for local variables. Maximum is 327680 bytes.

Serial Bluetooth WORDS --(Requires v7.0.6.5+)

esp_bt_dev_get_address	(-- a)	addr of 6 byte mac address
Serial Bluetooth		
SerialBT.new	(-- bt)	Allocate new BT object
SerialBT.delete	(bt --)	Free BT object
SerialBT.begin	(localname ismaster bt -- f)	
SerialBT.end	(bt --)	
SerialBT.available	(bt -- f)	
SerialBT.readBytes	(a n bt -- n)	
SerialBT.write	(a n bt -- n)	
SerialBT.flush	(bt --)	
SerialBT.hasClient	(bt -- f)	
SerialBT.enableSSP	(bt --)	
SerialBT.setPin	(z bt -- f)	
SerialBT.unpairDevice	(addr bt -- f)	
SerialBT.connect	(remotename bt -- f)	
SerialBT.connectAddr	(addr bt -- f)	
SerialBT.disconnect	(bt -- f)	
SerialBT.connected	(timeout bt -- f)	
SerialBT.isReady	(checkMaster timeout -- f)	Default checkMaster=false, timeout=0

Forth Code: Slave

```
0 Value BT0 Value bcounter 0 Value buffer
80 allocate drop to buffer

SerialBT.new to BTO
\ Returns a handle (address) to the Bluetooth (BTO)

z" ESP32-BT-Slave" 0 BTO SerialBT.begin \ Return a Flag
\ The slave unit will be visible as 'ESP32-BT-Slave'

\ Check to see if there is any incoming data available
BTO SerialBT.available ( BTO – n )

\ If there is:
Buffer 80 BTO SerialBT.readBytes to bcounter
```

All outgoing text must have CRLF, 1310 (0D0Ah) on the end of the string

Here is one way to accomplish it:

This will replace the XX on the end with 1310.

```
s” AcknowledgeXX” 2dup + 2 - 13 swap c!
2dup + 1 - 10 swap c!
BTO serialbt.write ( a n BTO – n )
drop 20 ms BTO SerialBT.flush ( BTO -- )
```

Forth Code: Master

```
0 Value BT0 Value bcounter 0 Value buffer
80 allocate drop to buffer

SerialBT.new \ Returns a handle to the Bluetooth (BTO)

z" ESP32-BT-Master" 1 BTO SerialBT.begin \ Returns a
Flag
\ The Master unit will be visible as 'ESP32-BT-Master'

\ Connect to Slave Unit
 z" ESP32-BT-Slave" BTO SerialBT.connect ( a BTO – f )
 IF ." Connection Successful" cr Else ." Failure" cr Then
 50 ms 3000 BTO SerialBT.connected drop

\ Send Command to Slave unit
```

s” Any Message you want XX”

All outgoing text must have CRLF, 1310, 0D0Ah, on the end of outgoing text strings

2dup + 2 - 13 swap c!
2dup + 1 - 10 swap c!
BTO serialbt.write (a n BTO – n)
drop 20 ms BTO SerialBT.flush

\ Check to see if there is any incoming data available
BTO SerialBT.available

\ If there is
Buffer size BTO SerialBT.readBytes to bcounter

//

Timing and Handshaking

Automated sending and receiving of info between 2 ESP32 units via Bluetooth is subject to timing delays. I determined for my 2 units the time from sending a command out and receiving an acknowledgement from the receiving unit was a full 2 seconds (2060 ms).
This is a long time in computers cycles but needs to be taken in to consideration. Sending a command, receiving and acknowledgement and then receiving a response to the command will take 4 full seconds.

main
BTO = 1073549576
ESP32-BT-Slave is Visible
Awaiting Connection
Incoming Data - Master Calling Slave Unit
Acknowledgement Sent
Incoming Data - Master Calling Slave Unit
Acknowledgement Sent
Incoming Data - Master Calling Slave Unit
Acknowledgement Sent
Incoming Data - Master Calling Slave Unit
Acknowledgement Sent
Incoming Data - Master Calling Slave Unit
Acknowledgement Sent
Ok

main
BTO = 1073542752
ESP32-BT-Master is Visible
Connecting
Successful
Incoming Data - Acknowledged
Incoming Data - Acknowledged
Incoming Data - Acknowledged
Incoming Data - Acknowledged
Incoming Data - Acknowledged
ok