ESP32/MATLAB Communication Code

A getting started guide

Overview

List of core code files

ESP32	MATLAB	Description
Message.h	Message.m	Implements a common high level message data class that keeps track of information before and after transmission.
SppBluetooth.h	SppBluetooth.m	Serial Bluetooth communication handler class. It takes care of message encoding, sending and receiving. *Note that the ESP code checks for new data, while Matlab uses an event driven structure to reduce processing jitter.*
TaskInterface.h		Implements an ESP32 task interface. It dictates what functions and variables are required for a task class.
Main.cpp		Takes care of all practical Bluetooth/scheduling/task call details.
board_type.h		Interface for setting and receiving the board type (e.g, Motor Controller) from the eeprom memory.
Sheduler.h		Implements a basic scheduler.

List of example files

ESP32 Demo Files	Description
tasks/DemoTask.h	Minimum viable implementation of a task.
tasks/DemoMotorControllerTask.h	Implementation of a simulated motor controller.
tasks/ DemoSensorBandTask.h	Implementation of a simulated sensor band.

High frequency signal sampling

Bluetooth typically have a polling rate of 125 Hz[1] where it processes incoming or outgoing data. This makes it impossible to send and process signals at 1 kHz. To remedy this, these signals are aggregated and transmitted at a lower frequency, e.g., at 50 Hz. I recommended to send signals with a low frequency content at a low frequency to reduce the bandwidth requirement and ultimately reduce variation in network latency. For implementation purposes, the aggregated high frequency samples are transmitted every x samples. It looks like this:

Transmission Number	1			2			3			4		
High Frequency Samples	1	2	3	4	5	6	7	8	9	10	11	12
Low Frequency Samples	1	1	1	4	4	4	7	7	7	10	10	10

Table 1 – Example of sending the high frequency samples every 4th sample. Note that the low frequency samples remain constant throughout a transmission period.

The low frequency transmission rate is calculated by:

[1] I've got this number from Bluetooth mouse specifications, but it corresponds nicely with my own findings.

$$f_l(f_h, x) = \frac{f_h}{x}$$

ESP32 Bluetooth name

Each ESP32 have a permanent and unique 12 hexadecimal MAC address that is incorporated into the Bluetooth network name. The mac address is pre-appended to “@Exo-Aider”. Some examples of network names

- 009ABBE350CC@Exo-Aider
- 60FB9912CFA4@Exo-Aider
- 74E80B12CFA4@Exo-Aider

ESP32 Tasks

An ESP32 task class must inherit the *TaskInterface* class. It contains a few metadata variables that must be set during the *initialization* function.

Field	Type	Description	Example
description	String		“Motor Controller”
high_frequency_sample_names	List of Strings	Names of the high frequency samples.	{“EMG1”, “EMG2”}
low_frequency_sample_names	List of Strings	Names of the low frequency samples.	{“torque”, “IMUx”, “FSR1”}

The ESP32 task metadata is extracted by MATLAB and used to keep track signals. For sample implementations, see:

- *DemoTask.h*
- *DemoMotorControllerTask.h*
- *DemoSensorBandTask.h*

To make a task available, it must be registered by the *“get_potential_tasks”* function in the *“tasks/task_list.h”* file.

The Message class

A message contains the following information:

Field	Type	Description
Command	String	Let’s the receiver know what to do with the message.
Numbers	List of floating-point numbers	Number arguments.
Strings	List of strings	String arguments.

Build in ESP32 commands

See main.cpp for the implementation of the different commands. An overview of the most useful arguments is provided here:

Command	Description/uses
ping	Used to check if the MATLAB-ESP32 connection is working.
set_board_task_name **get_board_task_name**	Sets/gets the board task name. This is the name of the active task. The board must be restarted for any changes to take effect.
restart	Restarts the board.
get_sheduler_periods_behind	Gets how many samples the scheduler is behind. Useful for debugging purposes.

set_sample_frequency **get_sample_frequency**	Sets/gets the high frequency sampling frequency.
set_send_signals_ratio **get_send_signals_ratio**	Sets/gets x from the High frequency signal sampling section.
set_send_signals **get_send_signals**	Sets/gets if the high and low frequency signals are being transmitted from the ESP32 to Matlab.
get_lf_signal_names	Gets a list of the low frequency sample names.
get_hf_signal_names	Gets a list of the high frequency sample names.
get_task_description	Gets the task description.
relay	Relays back whatever is send to the ESP32. Useful for determining if the communication protocol works.

Common Use Cases

Getting ESP32 configuration information

On boot the ESP32 transmits its metadata through the serial port. Sample information

```
Initializing...
 * Board task name: "demo_motor_controller"
 * Bluetooth name: "009ABBE350CC@Exo-Aider"
 * 4 potential tasks: ["demo_left_sensor_band", "demo_motor_controller",
        "demo_right_sensor_band", "demo_task"]
 * 23 low frequency signals: ["n", "t", "u", "y", "r", "s3", "s4", "s5", "s6",
        "s7", "s8", "s9", "s10", "s11", "s12", "s13", "s14", "s15", "s16", "s17",
        "s18", "s19", "s20"]
 * 0 high frequency signals: []
Initialized!
```

Adding a new ESP32 task (Motor Controller, Sensor Band, etc.)

1. Create a new class instance that inherent the *TaskInterface* class. See the demo files in “tasks/_Demo*.h” for information._
2. Add an instance of the class to the *get_potential_tasks* function in the “tasks/task_list.h” file.

Controlling the ESP32 from MATLAB

Coming soon…

Getting ESP32 samples from MATLAB

Coming soon…