Contents

1. Introduction

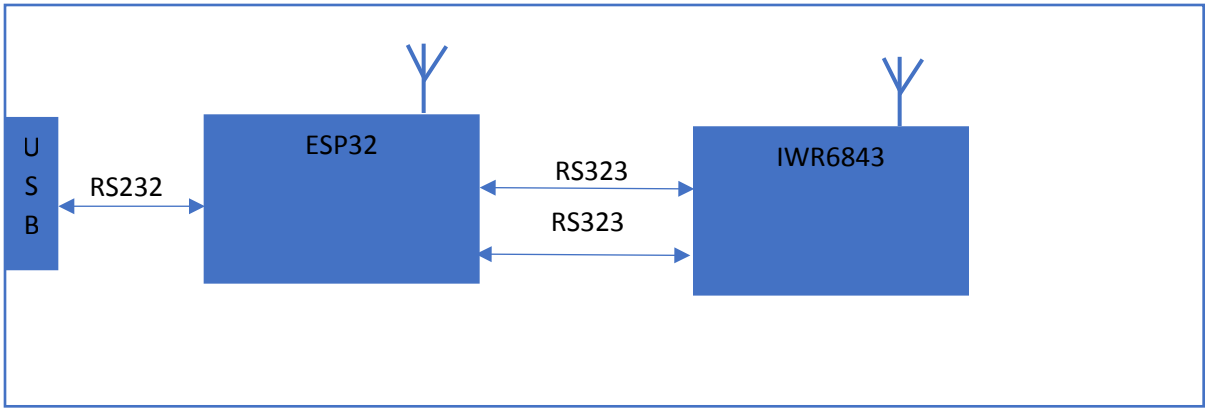
This project uses TI mmWave technology to detect the Human occupancy in the room. The following document captures the software high level design flow, implementation details, other dependencies and constraints. This includes bootloader design, the memory budget for all software components, message protocol and Hardware requirement to meet the SW design till some extent.

2. System Block Diagram

The blow is the overall block diagram of the system. It consists of IWR6843 mmWave sensor which data is given to ESP32 WiFi module to send to the HTTP server for further processing. Each block roles and responsibilities are explained below.

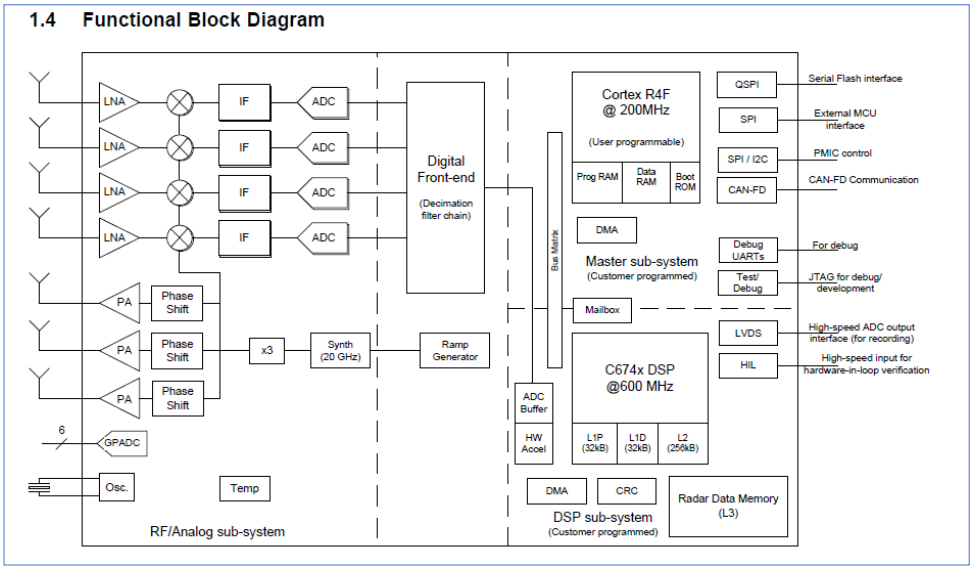
The communication channels are defined as below

1) mmWave Sensor to ESP Module – The detected objects detail from mmWave sensor will be sent to ESP via RS232.
2) ESP module to HTTP Host – The ESP module will send the final data to HTTP host via the WiFi AP.
3) External USB connector to ESP module/mmWave sensor – USB to Serial connection is used to program the boot loader S/W on ESP module and mmWave sensor.



3. mmWave Sensor

The below block diagram is captured from EVM (Evaluation Module) design for mmWave Sensor. The mmWave sensor is an integrated chip which has BSS (Baseband Radar module Sub-System), DSS (DSP Sub-System) and MSS (Master Sub-System). This module transceiver works on 60-64 GHz. The doppler signals are used to find the object with precise distance, speed and angle of arrival. The detected object details are sent out from the module as TLV (Type Length Value) packets for further processing.



The output message from mmWave sensor is as follows.

mmwDemo_output_message_header			TLV Data		
Field	Bytes	Value			Bytes
magicWord	8	0x0102030405060708	Type	MMWDEMO_OUTPUT_MSG_DETECTED_POINTS	4
version	4	SDK Version (0x03010102)	Length	n	4
totalPacketLen	4	Including header	Value	for(i=0; i<n; i+=16) {	
platform	4	0xA6843		float x;	4
frameNumber	4	Frame start track counter		float y;	4
timeCpuCycles	4	Time value (Need to check)		float z;	4
numDetectedObj	4	Num of detected objects		float velocity;	4
numTLVs	4	Num of TLVs (more than one TLV is possible for one object)		}	
subFrameNumber	4	(Need to check)			

The TLV data is followed by 40 bytes mmwDemo_output_message_header. After the message header the TLV data sent in loop. We are interested to check MMWDEMO_OUTPUT_MSG_DETECTED_POINTS tag. The object detected type is as follows. It is a 32-bit processor.

3.1. SYS/BIOS RTOS

SYS/BIOS is an advanced real-time operating system from Texas Instruments for use in a wide range of DSPs, ARMs, and microcontrollers. It is designed for use in embedded applications that

need real-time scheduling, synchronization, and instrumentation. SYS/BIOS provides a wide range of system services such as:

a. Preemptive, deterministic multi-tasking
b. Hardware abstraction
c. Memory management
d. Configuration tools
e. Real-time analysis

SYS/BIOS can be used on targets with a wide range of memory capacities. For example, a small application that includes timer, software interrupt, and task support can fit in less than 8 KB of Flash and less than 512 bytes of RAM, including stack space.

3.2. Secondary Bootloader (SBL)

The TI provided bootloader had support for functional mode where standalone boot is supported and support flashing mode where the user can program the image and debug mode is not useful for our case. The ESP module should know the updated image is available for flashing. The ESP should have a capability to force the mmWave mode to flashing mode through its GPIO. Below is the table for SOP (Sense On Power) line configuration for different modes of boot supported in mmWave sensor.

Table 1. SOP Lines and Boot Modes

SOP2 (P13)	SOP1 (P11)	SOP0 (J13)	Bootloader Mode and Operation
0	0	1	Functional mode The device bootloader loads the user application from the QSPI serial flash to the internal RAM and switches the control to it.
1	0	1	Flashing mode The device bootloader spins in loop to allow flashing of the user application (or the device firmware patch – supplied by TI).
0	1	1	Debug mode The bootloader is bypassed and the R4F processor is halted. This lets the user connect the emulator at a known point.

Flash programming by using Uniflash tool is possible by changing the switch to Position B (ESP/mmWave) on the base board. When the switch in Position B the external USB port is connected mmWave sensor programming interface, so we can use Uniflash tool to flash the image.

The mmWave sensor have secondary bootloader (SBL). The SBL bootloader architecture is as below.

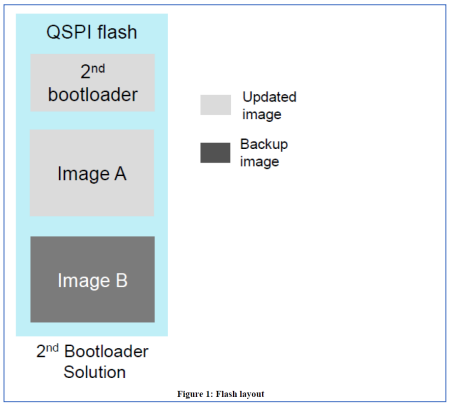


Figure 1: Flash layout

Baudrate: 115200
Data: 8 bit
Parity: None
Stop bit: 1 bit
Flow control: None

The SBL will be programmed by uniflash and and SBL will take care of normal image boot or image upgrade through ESP module. The SBL accepts XMODEM protocol.

3.3. Design Questions

Below are the questions for the SW design

1) Q: What data should be sent out to cloud or local http/https server through ESP module?
   A: Since the project is going to use local HTTP server there is no need at present for cloud. For the Human occupancy detection, the finally detected Human count should be sent to server for some specific defined period is finalized.
2) Q: What kind of security is required?
   A: At present the mmWave device doesn't have any security and it can be a future extension.
3) Q: Does the bootloader image should be signed?
   A: At present no need to sign the image and it can be a future extension.
   Q: We cannot change the default boot loader which is running in the chip. How can we do it?
   A: The default boot loader can be flashed only through the USB port on the product.
4) Q: Should we protect the final data sent to the server?
   A: At present no need to protect the data and it can be a future extension.
5) Q: How to track the object?
   A: It is up to the implementer.
6) Q: Can we finalize both ANDROID TEAM and FIRMWARE TEAM on the JSON file.

3.4. mmWave Configuration

The SW module has BSS, DSS and MSS api. Many configuration parameters are used to control different parameters of Radar which will give effect on the object detection. Proper study should be done on these configs and it needs to be adjusted for our operation to get the appropriate results.

4. ESP Module Software Design

The ESP module should have the below implementations.

1) Establish the WiFi connectivity to the local WiFi AP which has access to the HOST.
2) Control the mmWave sensor boot mode by sending the BREAK signal on RS232 interface.
3) Read the flash image file from server and flash mmWave sensor through the UART port by XMODEM protocol.
4) Read the received object detected command and sent to HOST through WiFi.

5. SW Estimation

Below is the approximate SW development estimation.

S.No	Task	Duration	Dependency	Deliverable	Status
1	SW High level Design document (HLD)	2 Weeks		HLD	
2	ESP32 Board bring up	1 Week	HW		
3	IWR6843 Board bring up	1 Week	HW		
4	IWR6843 Data analysis	2 Week			
5	Receive detected object data from IWR6843 and push as JSON data to HTTPS from ESP32	1 Week			
6	Communication protocol and messages for IWR6843	1 Week			
7	Communication protocol and messages for ESP32	1 Week			
8	Boot loader for IWR6843	2 Weeks			
9	Boot loader for ESP32	2 Weeks			
10	Fine tune algorithm if anything required	1 Week			
	Total	14 Weeks			