ECEN 5013
FINAL PROJECT REPORT

CAR SAFETY MECHANISM

BY: TEJAS SHANBHAG
DHARMIK THAKKAR

OVERVIEW OF THE PROJECT AND THE PROBLEM IT SOLVES

Technology has advanced to an optimum level with embedding sensors used for various purposes like security, safety, communication, etc. involved in our day to day activities.

The purpose of this project is to employ the concepts learnt in the course and implement a car safety mechanism using different sensors and using the Bluetooth Low energy for wireless communication. A specific set of sensors are integrated to detect certain conditions and alert the emergency contacts and emergency services in case of safety issues.

The sensors used here are the temperature sensor, accelerometer, capacitive sensor, and ambient light sensor.

The project solves the high risk of car accidents when the car is at a high speed and suddenly decelerates due to an obstacle or misjudgement on the part of the driver. In case of such a case, the services get an alert in case of an accident and immediate actions but for the safety part, air bags are deployed immediately to protect the driver. Also, it continuously monitors the machines for temperature changes which is supplied to the fire brigade if the car starts to catch a fire and the driver to alert them of sudden temperature elevations in the machinery. Negative temperatures can also be monitored. Thus, the driver can turn off his machine in case of elevated temperatures.

Another added specification is the use of an ambient light sensor for automatic light control. The light is detected from the surroundings with the help of the sensors and the intensity of the headlights is controlled in accordance thus optimising power.

A touch sensor which is a capacitive touch sensor in this case, is used for turning off the system whenever not required to save power. However, the accelerometer output is not affected because that is necessary and must be on always.

There are some high-risk aspects as well. Getting an alert after the accident can be an advantage but immediate help to the driver cannot be provided. Also, negligence on the part of the driver on temperature monitoring can lead to fatal casualties since temperature rise need not certainly be a problem.

Key components used

- ATMEL ALSAMB11 as the master to communicate via BLE.
- Leopard Gecko STK3600 starter kit as the sensor hub.
  - Internal temperature sensor
  - Capacitive touch sensor using LESENSE
  - Ambient light sensor using TSL2561
- 3 axis accelerometer MMA8452Q

- On-board ADC for conversion of temperature values.
- DMA for transferring the ADC values to the transmitting buffer.
- I2C interface of the sensor hub for communication with the two external sensors.
- Low Energy UART interface of the Gecko to communicate with the SAMB11
- UART of the SAMB11
- ALMEL SMART APPLICATION for BLE

Software Trigger points or events

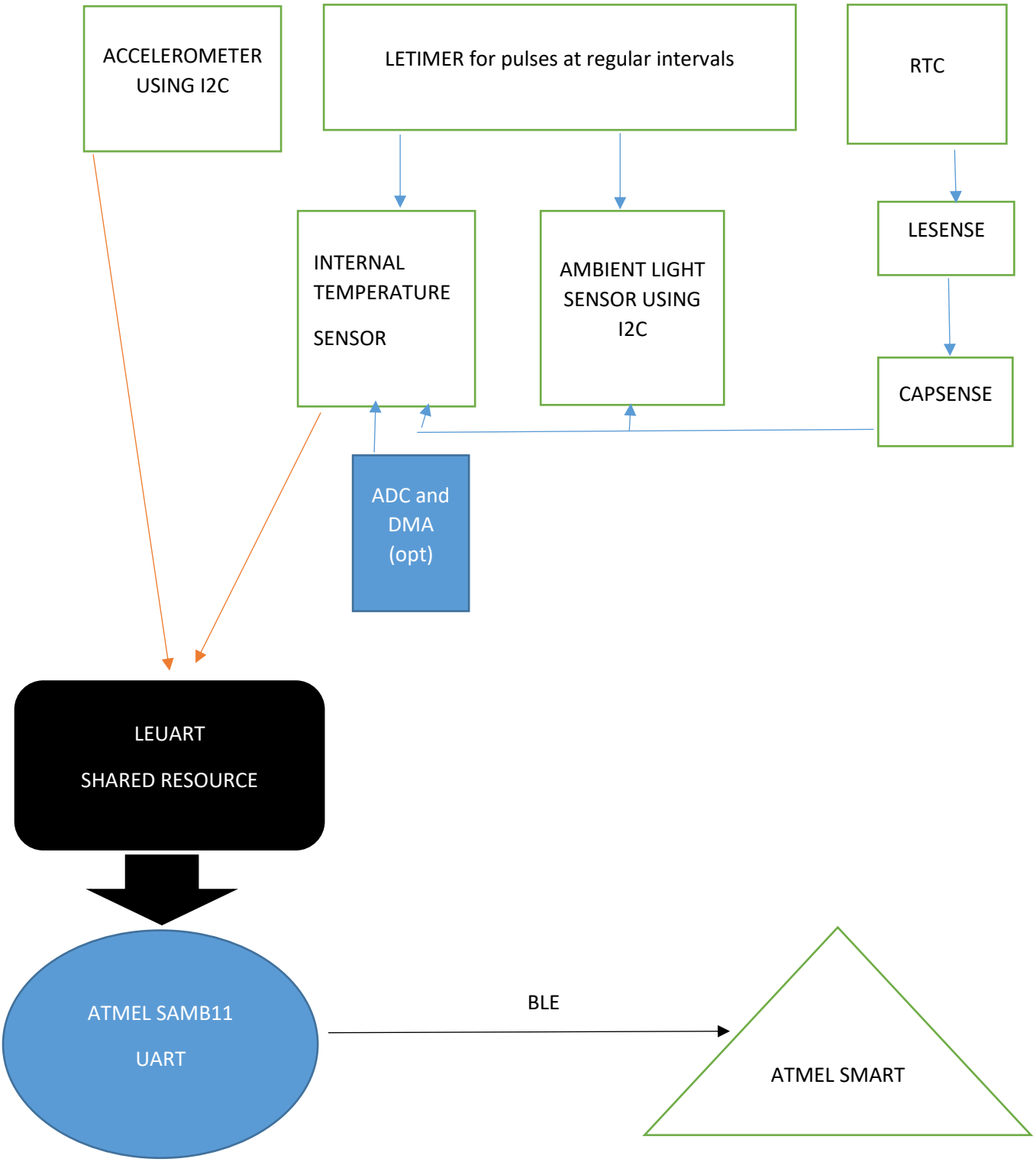
- The letimer receives an interrupt every 5 sec where it does its specific functions like turning on ADC for temperature conversions. Every time data must be sent, the ADC gets an interrupt.
- The capacitive sensor is controlled via LESENSE and a trigger is received every time the capacitive sensor is pressed. The debouncing issue is solved here.
- For the ambient light sensor, we give lower and higher thresholds. As a result, whenever the ambient light crosses these thresholds, an interrupt is generated.
- For the accelerometer, we use it in transient mode. Thus, whenever the acceleration crosses certain predefined thresholds, an interrupt is generated.
- On the atmel side, we check whether we are getting an alert signal or temperature signal and thus the output is decided accordingly.

List of commands used

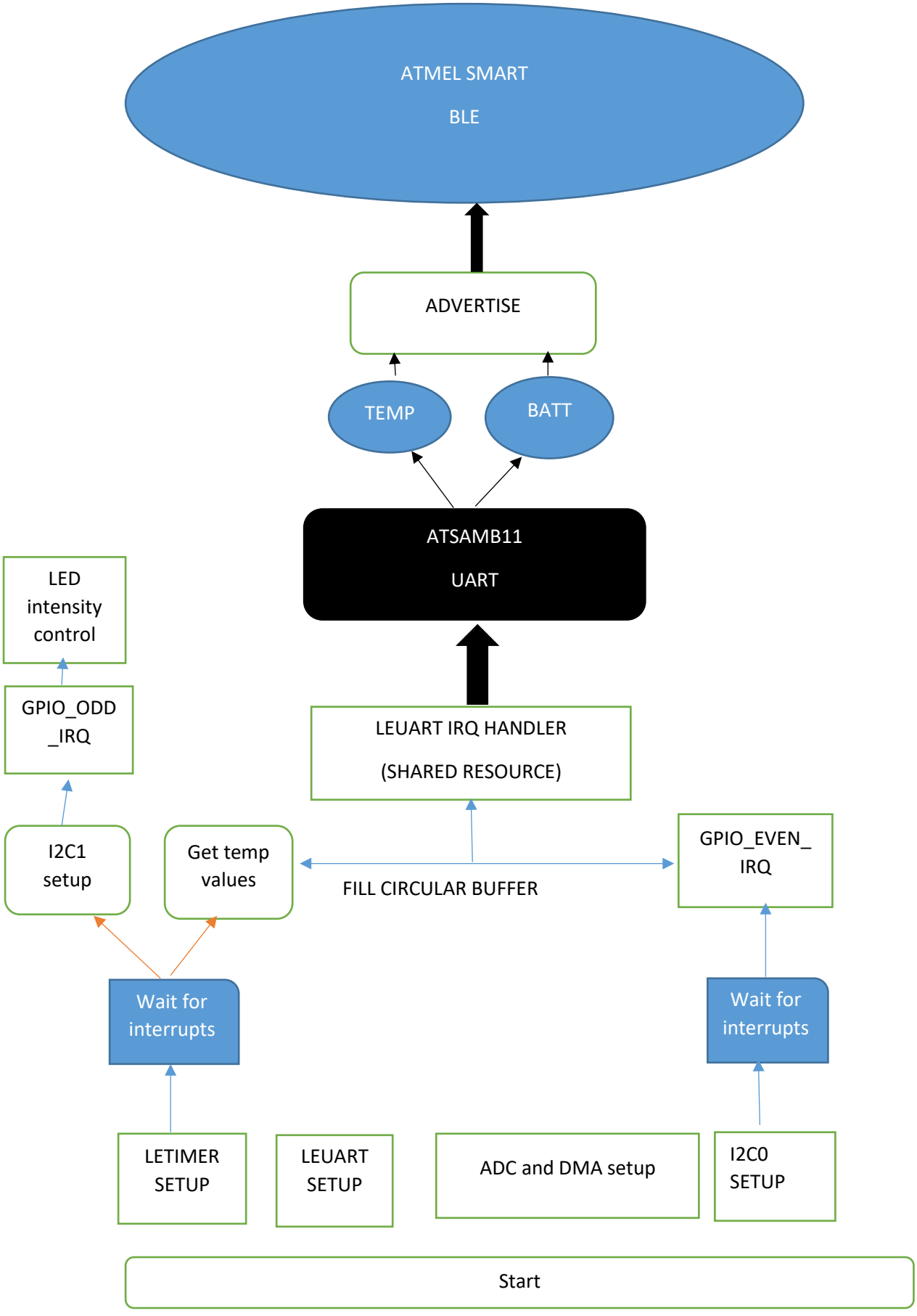
The inputs received via the LESENSE are LETIMER_Enable(LETIMER0, false) and LETIMER_Enable(LETIMER0, true) to turn ON and OFF the letimer since all the other peripherals are working on it.
For both the I2Cs we have made read and write drivers to write and read from the I2c registers.
We get an interrupt whenever a jerk is given to the LED.
For LED control, we use

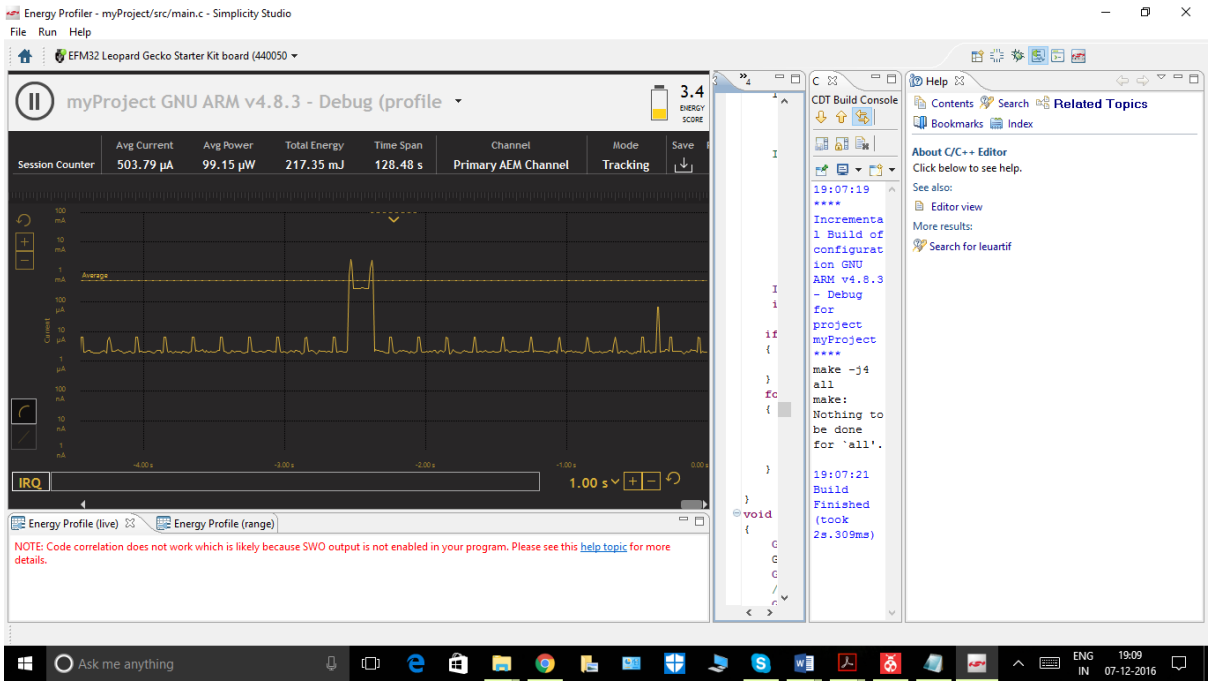
GPIO_PinModeSet(LEDport, LEDpin, *gpioModePushPull*, 1);
GPIO_PinModeSet(LEDport, LEDpin, *gpioModePushPull*, 0);
GPIO_PinModeSet(LEDport, LEDpin, *gpioModePushPullDrive*, 0);
GPIO_PinModeSet(LEDport, LEDpin, *gpioDriveModeLow*, 1);

HARDWARE BLOCK DIAGRAM

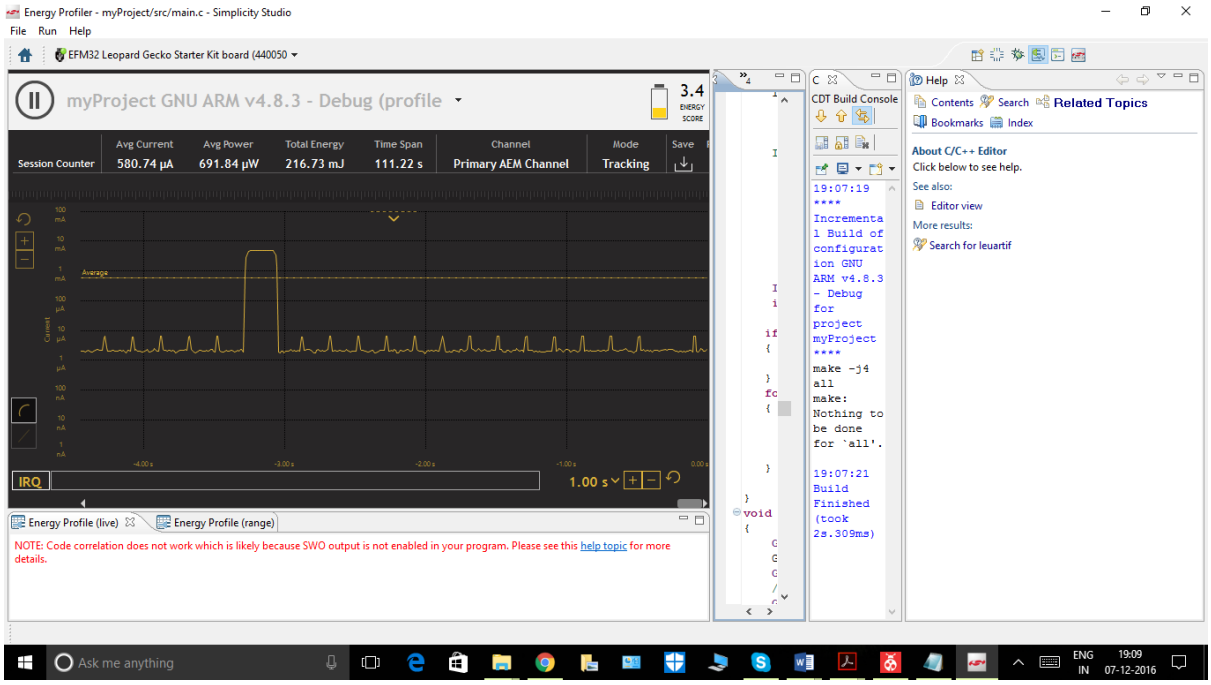


SOFTWARE FLOW ORGANIZATIONAL CHART

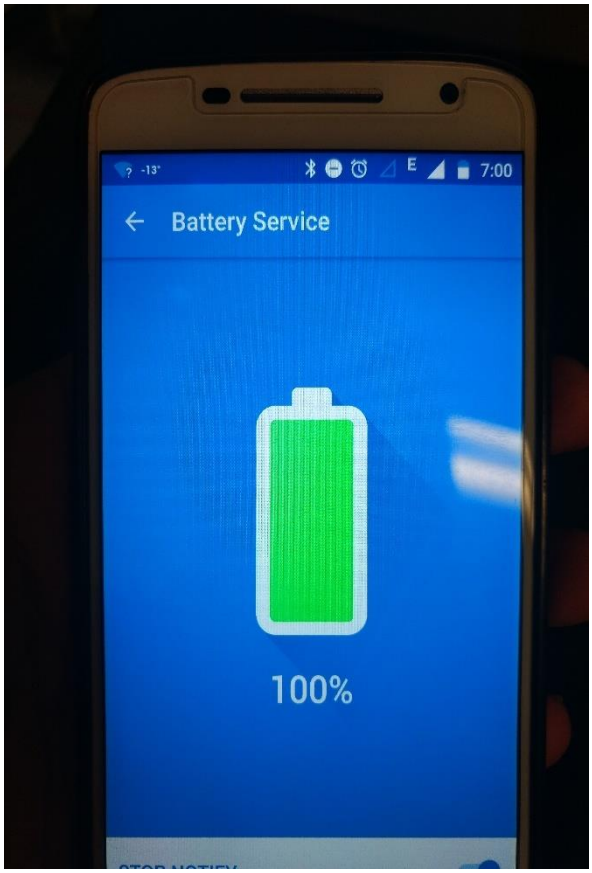




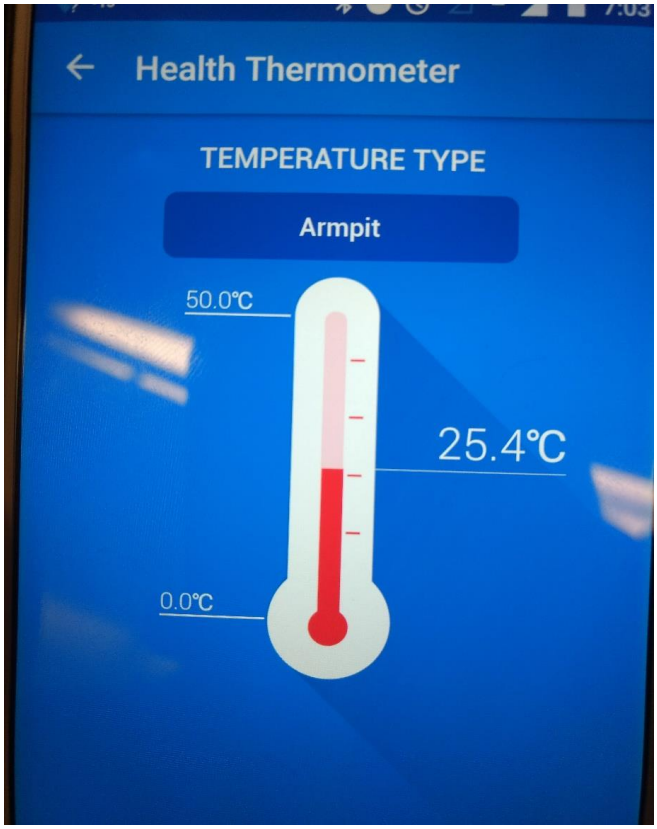
Accelerometer working in EM2



LESENSE working in EM2



Battery profile working



Temperature profile working

Plan development schedule	Date	Status
Data sent via circular buffer	Nov 1	Done
Working of sensors in Leopard Gecko	Nov 8	Working of temp sensor and capacitive sensor
LESENSE working independently	Nov 8	Done
Working of TSL2561	Nov 13	Working
Two profiles working and updated individually	Nov 15	Done
Working of add on sensor	Nov 25	Working
UART interfacing with SAMB11	Nov 27	done
Data sent via LEUART	Nov 29	Both data sent
Connection via Bluetooth for both profiles	Nov 29	Done
Both profiles working in main with updation	Dec 4	Done
LESENSE capsense integration with the main	Dec 6	Done

Problems faced while the making of the project:

1. Understanding the working of LESENSE was a major problem faced while using it.
2. Deciding whether using the ambient light sensor with lesense or with TSL2561 was an issue.
3. The major problem faced was getting the alert notification profile on ATMEL SMART since the phone alert profile worked fine independently but it couldn't be integrated with any other profile. This problem took a lot of time.
4. Since the accelerometer is always ON, it had to be given power. Hence, the GPIO power pin took a lot of energy not letting it go in EM2 or EM3.
5. One other challenge was using both the I2Cs simultaneously.

Summary of the project:

As a result, for all the modules,

The accelerometer was used in low power mode instead of the normal or high resolution mode since the performance does not degrade much. Also, we use the transient operation where we don't detect the motion but get an interrupt when an acceleration above a certain threshold is detected. This is when we get an alert. Hence it should be monitored continuously and should be independent of the LETIMER.
One of the problems above is solved by using the 3.3V pin on the gecko instead of the power pin.

Now for the ambient light problem, we can use the lesense for power saving, but since for the application, we need high accuracy for light sensing and give corresponding output, we use the TSL2561.

Also, the LESENSE is periodically configured via the RTC to avoid interference with the LETIMER.

Referring to the main problem, I decided to go with the battery profile since it was more convenient to show alert on the battery profile.

Hence, all the modules required for the project worked fine independently and together without interfering with one other to give a combined cohesive unit working on a prototype basis.

LESSONS LEARNT

- Using different sensors and peripherals both on board on of board in their different modes.
- Using different communication standards like I2C and UART.
- Using wireless communication like BLE and actually using it via the phone and showing multiple profiles working.
- Basic debugging and register level programming.

ESE DELIVERABLES

● **Modularity:** Entire firmware is divided into smaller modules (functions) specific to a peripheral. For example: LETIMER_init, LEUART_init, ADC_init, DMA_set_up, I2C_init etc. Furthermore, read and write function libraries are developed for communication with I2C based peripherals.

● **Testability:**

Component under test	Feature	Plan test	Definition of pass	Did it pass
Temperature sensor	Sensing absolute temperature in celcius	Run the code and send to the Samb11	Values updated and average value found	yes
Sensing negative temperatures	Sensing negative temperature in celcius and displaying it	Hard code the temperature or use a cold spray	Negative values received	yes
DMA working	Transferring the values of temperature	Debug mode	Buffer getting filled and average calculated	yes
Transfer without dma	Transferring the values of temperature	Debug mode	Buffer getting filled and average calculated	yes
Cap sensor	Touch sense	Touch the last section	LEDs toggling on each touch	Yes
Lesense cap sensor	Turning off temperature sensor	Touch the last section	No pulses found on the energy profiler and Values not	yes

			getting updated on the BLE	
Lesense cap sensor	Turning off ambient light sensor	Touch the last section	LED value not getting updated	yes
Lesense cap sense	Turning on all the peripherals	Touch again	LETIMER resynchronizes and temp pulses are received with ambient light being monitored	Yes
Ambient light	Detecting darkness	Absolute darkness	LED with full intensity	yes
Ambient light	Detecting moderate darkness	Little light	Led with high intensity	yes
Ambient Light Sensor	Detecting moderate brightness	More light	LED with low intensity	yes
Ambient Light Sensor	Detecting maximum brightness	Absolute brightness	LED off	YES
LEUART	Send temperature data	Check in leuartirq handler	Checked in rxdata via loopback mode	yes
LEUART	Send Acc data	Check in rxdata with loopback mode	Data sent	yes
LEUART	Sending both data via circular buffer correctly	Check the received buffer values	Data sent is received without corruption or both profiles working perfectly	yes
LEUART	Back to back transfers	Check battery profile	No alert received	yes
Accelerometer	Transient acceleration sensing	Jerk above a certain acceleration	Led toggling	Yes
Accelerometer	EM2 or EM3	Try entering EM2 and EM3 while using the I2C	Current reaching specific values and acc working	yes
Getting temp data via gecko	Values getting transfered	Run both the interfaces	Data on Atmel smart getting updated	Yes

Getting alerts	Battery profile	Jerk the accelerometer and send via UART	100% for a short pulse and back to 0%	Yes
Both profiles working simultaneously	Both profiles working	Running both interfaces and giving jerks	Values getting updated for both profile	yes

● **Efficiency –** Higher efficiency of the code is achieved through well-timed enabling and disabling of the peripherals and making critical sections of the code atomic (for example emergency alert condition).

● **Robustness –** Code robustness is tested for by giving various inputs like:

- Emergency conditions at various instants.

- Negative as well as positive temperature inputs.

● **Structures** – circular buffer.
instances of defined functions: i2c init, acmp_init, letimer_init, timer_init, dma_init, adc_init, leuart_init

● **Enumerations** – accelerometer addresses, accelerometer register values, buffer states, tsl_register_addresses.

● **Preprocessor Macros (Constants, Functions, and Compile Time Switches)** – DMA_use, TSL_switch, cirbuf, definitions for variables, threshold values, ports, adc configuration, uart configuration, i2c configuration. Functions usage, imputs and returns as commented in code.

● **Interrupts** - letimer_irq, adc_irq, gpio_irq, leuart_irq

● **Software Abstraction** – use of BLE smart application services.

● **DMA** – for adc_transfer

● **Communication & Messaging** – uart communication, with message encoding and decoding

● **Events** – DMA callbacks after transfer complete, too bright light_event, acceptable light event, low light, dark, sudden acceleration/deceleration, system on(cap sense)

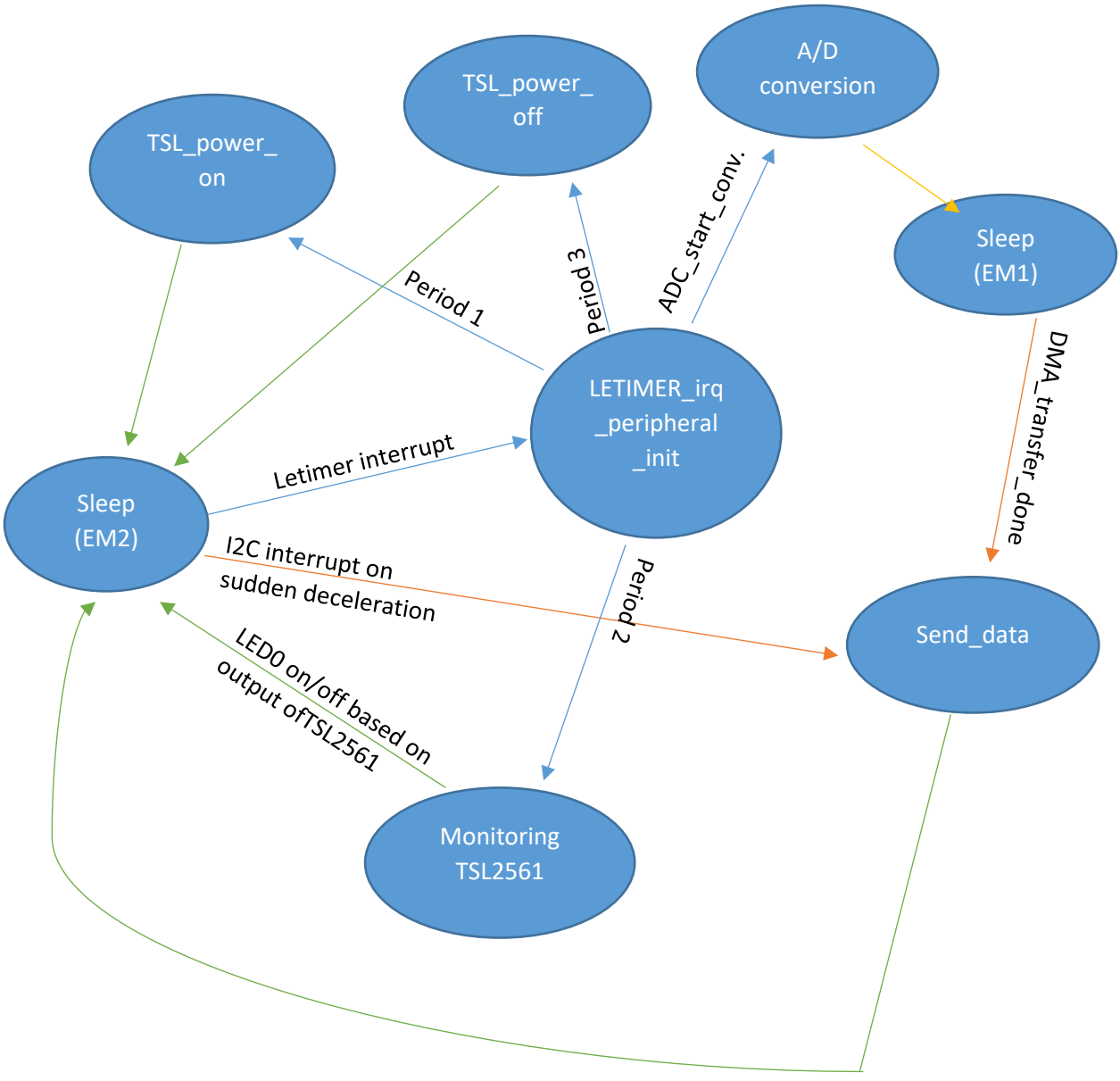
● **Buffers** – Circular buffer

● **Version Control** - https://github.com/dharmikthakkar/DHARMIK_ECEN5013.git
(In the branch project_4_rel)

● **Logging** – temperature and emergency alert data sent to ATSAMB11

● **Critical Sections** – leuart shared resource problem solved using circular buffer. Emergency sudden deceleration alert implemented using instantly enabling the leuart interrupt to send the alert condition.

- **State Machines –**



ADD ONS

Signal processing: use of ADC to convert analog data from onboard temperature sensor to digital. 12-bit resolution is used with a reference voltage of 1.25 volts. ADC module is triggered for start of conversion every 5 seconds (period defined using LETIMER). DMA transfer is used to transfer converted data to memory while letting CPU to sleep in EM1. After 500 such transfers, average temperature is evaluated for further analysis.

Low power: Entire system is optimized for minimum possible power consumption through use of:

- Low energy modes EM1(Sleep) and EM2(deep sleep)
- LETIMER, LESENSE (capacitive touch sensor), LEUART
- DMA for transferring ADC data to memory
- Load power management for TSL2561
- Enabling peripherals only for required duration and then disabling them for rest of the operation.
- Accelerometer (MMA8452Q) operated in low power mode.