IUPUI | SCHOOL OF ENGINEERING AND TECHNOLOGY

A PURDUE UNIVERSITY SCHOOL

Indianapolis

ECE 59500 – Embedded Autonomous Systems

Project Report

CAN Protocol Based Temperature Monitoring

Group Members:

Archit Muchhal
Tanmay Parashar
Sreeram Venkitachalam

Under the Guidance of
Prof. Dr. Mohamed El-Sharkawy

Spring 2018

CONTENTS:

1. Introduction

Overheating of engines has always been a factor that companies manufacturing automobiles have always given utmost priority to as it can have hazardous impact on the vehicle in the long run.

Anything that lessens an engines capacity to absorb, transfer and dissipate heat has the capability of overheating an engine. Overheating in engines is generally caused because of factors like having low coolant levels, faulty thermostat, low motor oil levels and many others. If in case any of such thing arises, the vehicle should not be used and should be serviced to avoid failure of various engine components.

In our project we check this overheating condition using temperature sensors. If the temperature is above an optimized level, the engine would not turn on. Also, if in motion such an overheating condition arises, the engine would shut down and the vehicle would come to a stand-still hence avoiding damage to the vehicle components.

We are using a temperature sensor (TMP-36) which is a 8-bit analog sensor which would sense and transmit the analog temperature values to the S32K144 board (Node A). These values would be sent to the other S32K144 board (Node B) using CAN protocol to which the motor is connected.

All the calculations are done at the Node A end. If the temperature values received at the Node B end are above the optimized values, the motor would not start and if they are below the optimized values, the motor would start.

The purpose of this project is to prevent the engine from turning on if an overheating condition arises and to shut off the engine if an overheating condition arises.

This would prevent further damage to vehicle components and would ensure maximum efficiency of the car.

<u>CAN Protocol:</u>

A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based protocol designed originally for multiplex electrical wiring within automobiles to save on copper but is also used in many other contexts.

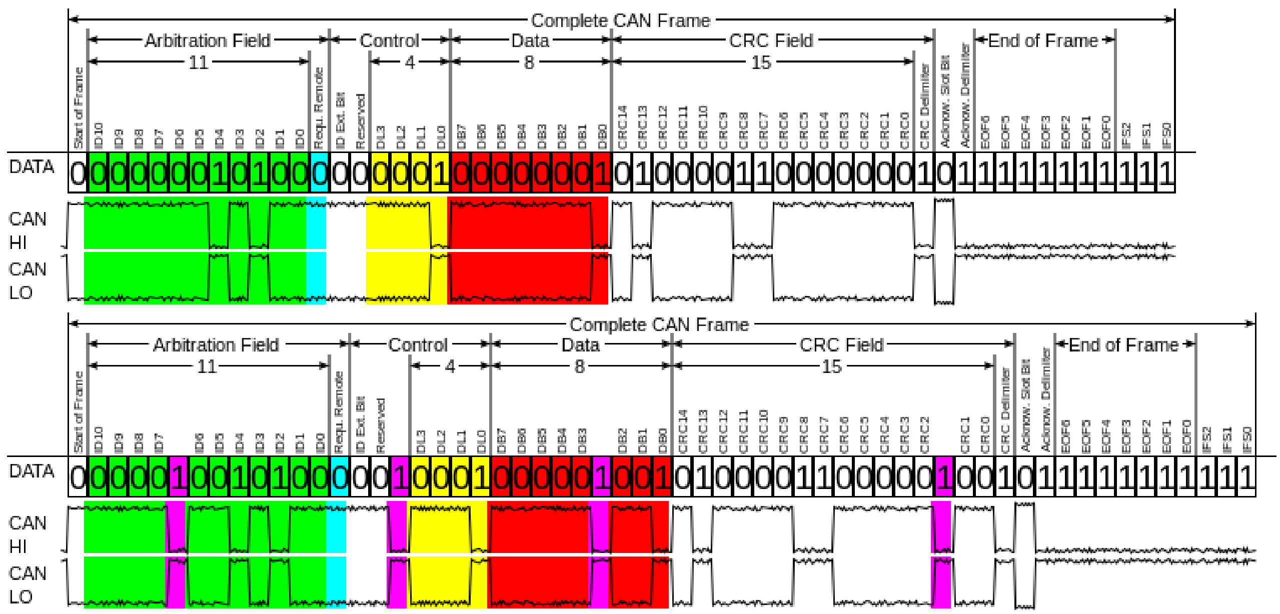


Figure 2: CAN Data Fram

2. Implementation

On Board Processing

Initially when (SW=3) is pressed the motor should turn on. The temperature sensor (TMP-36) is heated using a Heat gun. These values are sent to the NXP (S32K144-EVB). The corresponding temperature values can be viewed using (FreeMaster). The NXP (S32K144-Node A) will send the analog temperature values to the NXP (S32K144-Node B) using the CAN bus. Node A will do the necessary calculations and will convert the analog temperature values into degree Celsius (^{o} C) for display. If the received temperature value > 60 ^{o} C, the motor would stop running and wait for (10 seconds) and will start again if the temperature goes below 60 ^{o} C. If not, the motor goes into failure mode.

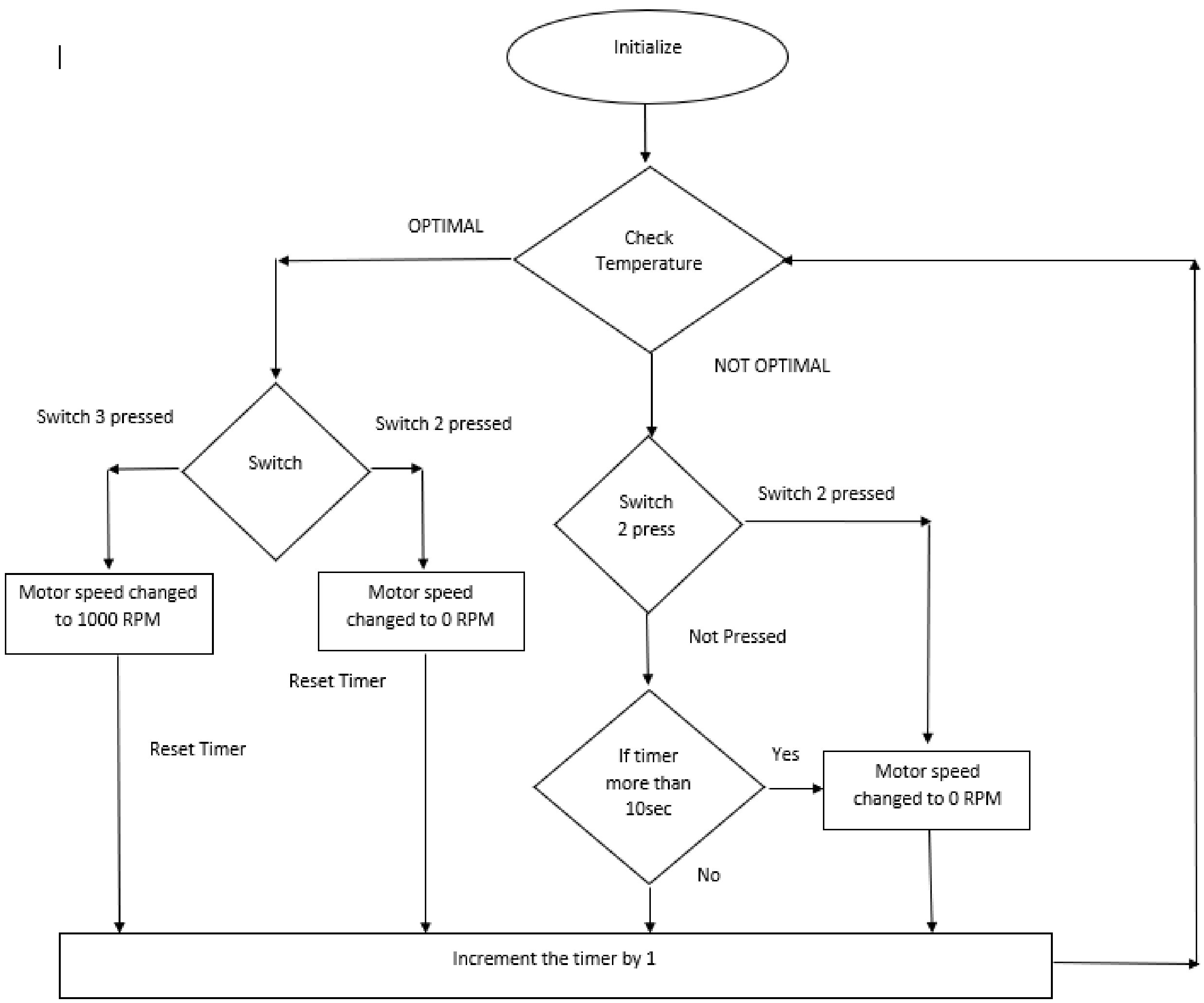


Figure 3: Flow Chart

Simulink Code Blocks:

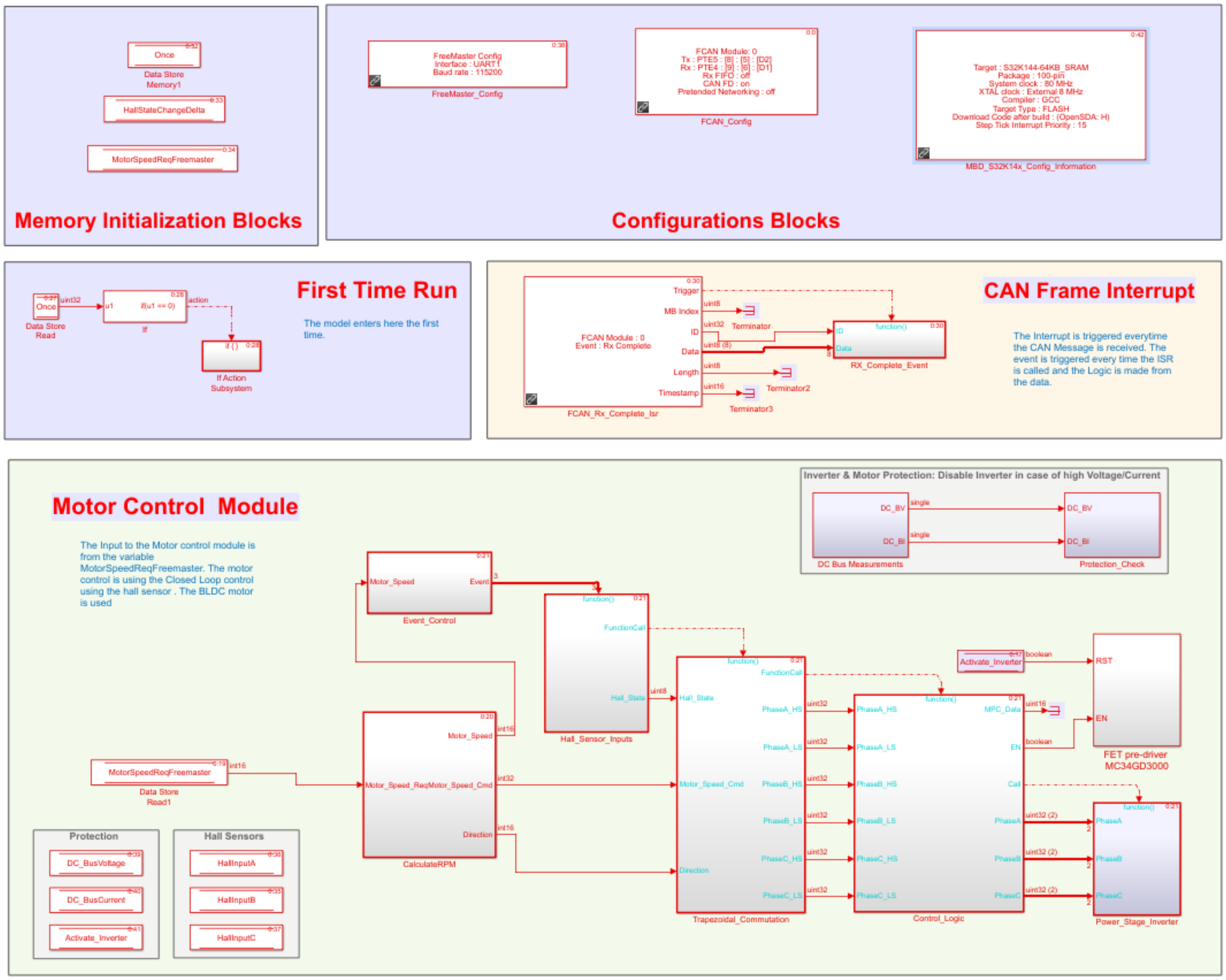


Figure 4: SIMULINK blocks for Node B

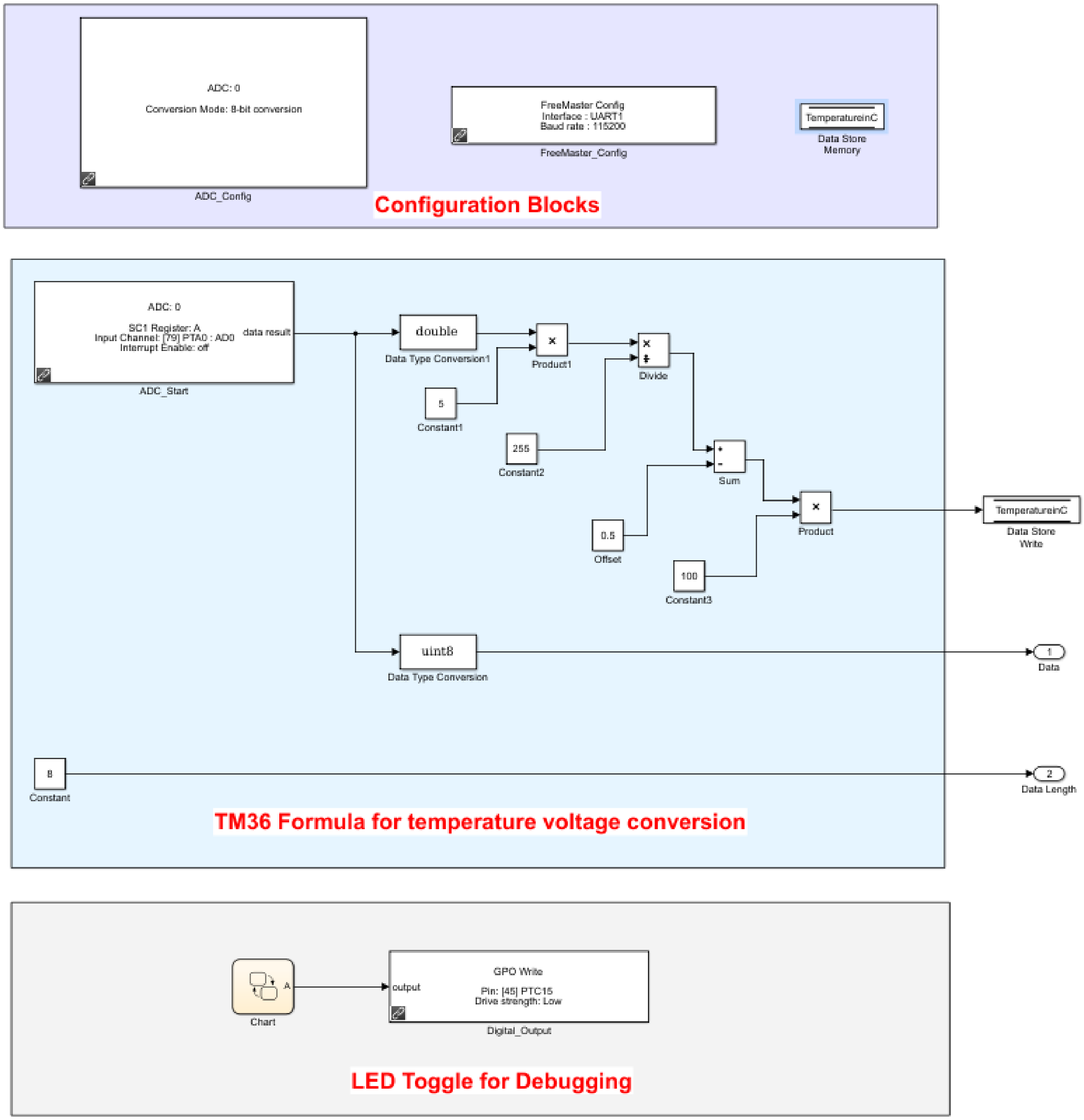


Figure 5: SIMULINK blocks for Node A (Formula block)

3. Hardware

The hardware required for this project are NXP's FRDM S32K144 boards, Motor shield, Motor, CAN bus, Temperature sensor (TMP-36)

Figure 6:LINIX Motor

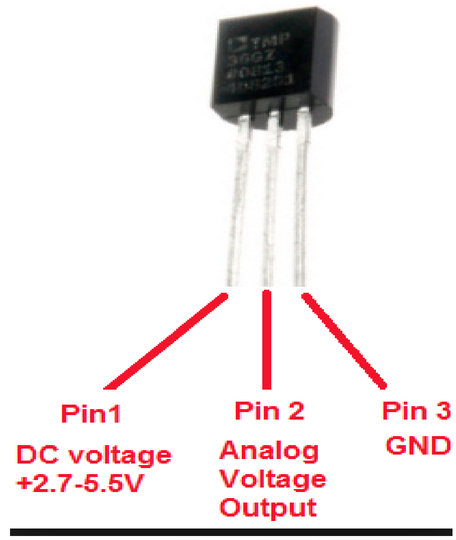


Figure 7: Pinout diagram for TMP 36

Figure 8: Motor Control Shield

4. Operation

Initially, the motor is off which signifies that the engine is off. Motor on signifies that the engine is on. Once switch 3 is pressed the motor will turn on provided that the engine temperature is nominal. The TMP-36 is a 8-bit analog temperature sensor which will sense the temperature of the engine and will send corresponding temperature values to the controller (S32K144-Node A). The S32K144 controller receives the analog temperature values from the temperature sensor and it converts it to degree Celsius for display purposes. Node-A of the controller will send the analog temperature values to (S32K144-NodeB) using CAN protocol. The Node B is responsible for controlling the motor. If the received temperature values >60 degrees Celsius then the motor will monitor the temperature readings for 10 seconds. If in that period the temperature drops below 60 degrees, the motor will keep running. If the temperature does not drop below 60 degrees, the motor will turn off.

Requirements

Software

- MBDT/Simulink
- FreeMaster

Hardware

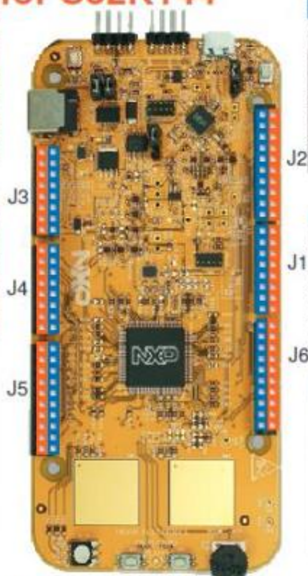
- S32K144 (Node A &B)
- CAN Bus
- DevKit-MotorGD
- TMP-36 Temperature sensor
- DC Motor
- LINIX Motor
- Battery (12&15V)
- USB Cable

Header/Pinout Mapping for S32K144

PIN	PORT	FUNCTION	J3	PIN	PORT	FUNCTION
J3-02	PTB6*	GPIO		J3-01		VIN
J3-04	PTB7*	GPIO		J3-03		IOREF
J3-06	PTE0	GPIO		J3-05	PTA5	RESET
J3-08	PTE9	GPIO		J3-07		3V3
J3-10	PTC5	GPIO		J3-09		5V
J3-12	PTC4	GPIO		J3-11		GND
J3-14	PTA10	GPIO		J3-13		GND
J3-16	PTA4	GPIO		J3-15		VIN

PIN	PORT	FUNCTION	J4	PIN	PORT	FUNCTION
J4-02	PTC7	GPIO		J4-01	PTD4	ADC0
J4-04	PTC6	GPIO		J4-03	PTB12	ADC1
J4-06	PTB17	GPIO		J4-05	PTB0	ADC2
J4-08	PTB14	GPIO		J4-07	PTB1	ADC3
J4-10	PTB15	GPIO		J4-09	PTA6/PTE11/PTA2	ADC4
J4-12	PTB16	GPIO		J4-11	PTC0/PTE10/PTA3	ADC5
J4-14	PTC14	GPIO		J4-13	PTE2	ADC6
J4-16	PTC3	GPIO		J4-15	PTE6	ADC7

PIN	PORT	FUNCTION	J5	PIN	PORT	FUNCTION
J5-02	PTE16	GPIO		J5-01	PTA15/PTD11	ADC8
J5-04	PTE15	GPIO		J5-03	PTA16/PTD10	ADC9
J5-06	PTE14	GPIO		J5-05	PTA1	ADC10
J5-08	PTE13	GPIO		J5-07	PTA0	ADC11
J5-10		VDD		J5-09	PTA7	ADC12
J5-12		GND		J5-11	PTB13	ADC13
J5-14	PTE1	GPIO		J5-13	PTC1	ADC14
J5-16	PTD7	GPIO		J5-15	PTC2	ADC15
J5-18	PTD6	GPIO		J5-17	NC	GPIO
J5-20	PTC15	GPIO		J5-19	NC	N/A



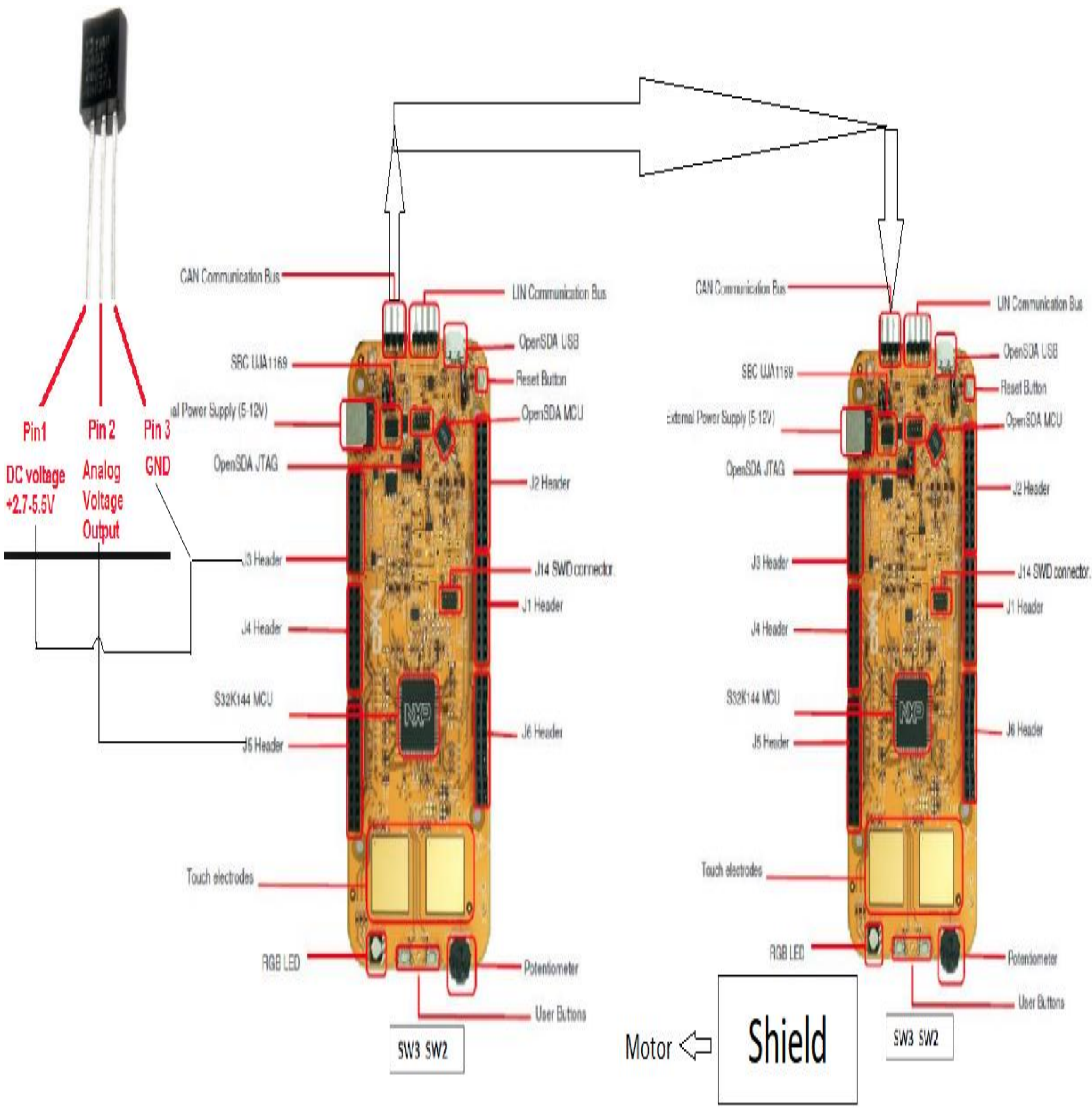
PIN	PORT	FUNCTION	J2	PIN	PORT	FUNCTION
J2-19	PTE10/PTA3	D15/I2C_CLK		J2-20	NC	GPIO
J2-17	PTE11/PTA2	D14/I2C_SDA		J2-18	NC	GPIO
J2-15		ANALOGUE REF		J2-16	PTA14	GPIO
J2-13		GND		J2-14	PTE7	GPIO
J2-11	PTB2	D13/SPI_SCK		J2-12	PTC13	GPIO
J2-09	PTB3	D12/SPI_SIN		J2-10	PTC12	GPIO
J2-07	PTB4	D11/SPI_SOUT		J2-08	PTE8	GPIO
J2-05	PTB5	D10/SPI_CS		J2-06	PTD0	GPIO
J2-03	PTD14	D9/PWM		J2-04	PTD16	GPIO
J2-01	PTD13	D8/PWM		J2-02	PTD15	GPIO

PIN	PORT	FUNCTION	J1	PIN	PORT	FUNCTION
J1-15	PTC11/PTE8	D7		J1-16	PTE3	GPIO
J1-13	PTC10/PTC3	D6		J1-14	PTD3	GPIO
J1-11	PTB11	D5		J1-12	PTD5	GPIO
J1-09	PTB10	D4		J1-10	PTD12	GPIO
J1-07	PTB9	D3		J1-08	PTD11	GPIO
J1-05	PTB8	D2		J1-06	PTD10	GPIO
J1-03	PTA3	D1		J1-04	PTA17	GPIO
J1-01	PTA2	D0		J1-02	PTA11	GPIO

PIN	PORT	FUNCTION	J6	PIN	PORT	FUNCTION
J6-19	PTA9	D14		J6-20	PTE4	GPIO
J6-17	PTA8	D15		J6-18	PTE5	GPIO
J6-15	PTE12	D16		J6-16	PTA12	GPIO
J6-13	PTD17	D17		J6-14	PTA13	GPIO
J6-11	PTC9	D18		J6-12		GND
J6-09	PTC8	D19		J6-10		VDD
J6-07	PTD8	D20		J6-08	PTC16	GPIO
J6-05	PTD9	D21		J6-06	PTC17	GPIO
J6-03	PTD2	GPIO		J6-04	PTD3	GPIO
J6-01	PTD0	GPIO		J6-02	PTD1	GPIO

Arduino compatible pins
NXP pins

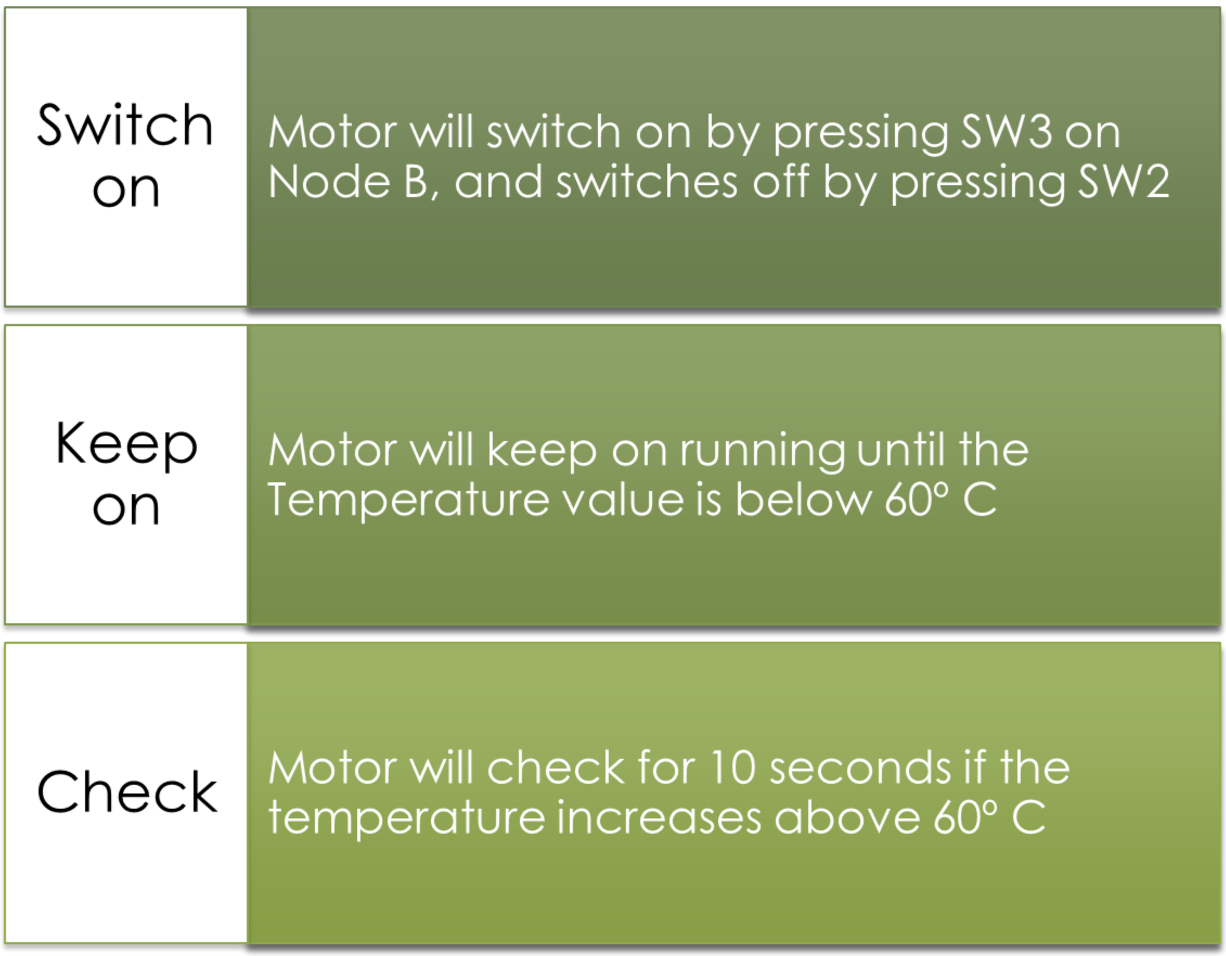
*0ohm resistor is not connected



Pin1
DC voltage
+2.7-5.5V
Pin 2
Analog
Voltage
Output
Pin 3
GND
CAN Communication Bus
LIN Communication Bus
OpenSDA USB
SBC UJA1169
Reset Button
OpenSDA MCU
OpenSDA JTAG
J2 Header
J3 Header
J14 SWD connector.
J1 Header
J4 Header
S32K144 MCU
J6 Header
J5 Header
Touch electrodes
RGB LED
Potentiometer
User Buttons
SW3 SW2
External Power Supply (5-12V)
Motor
Shield

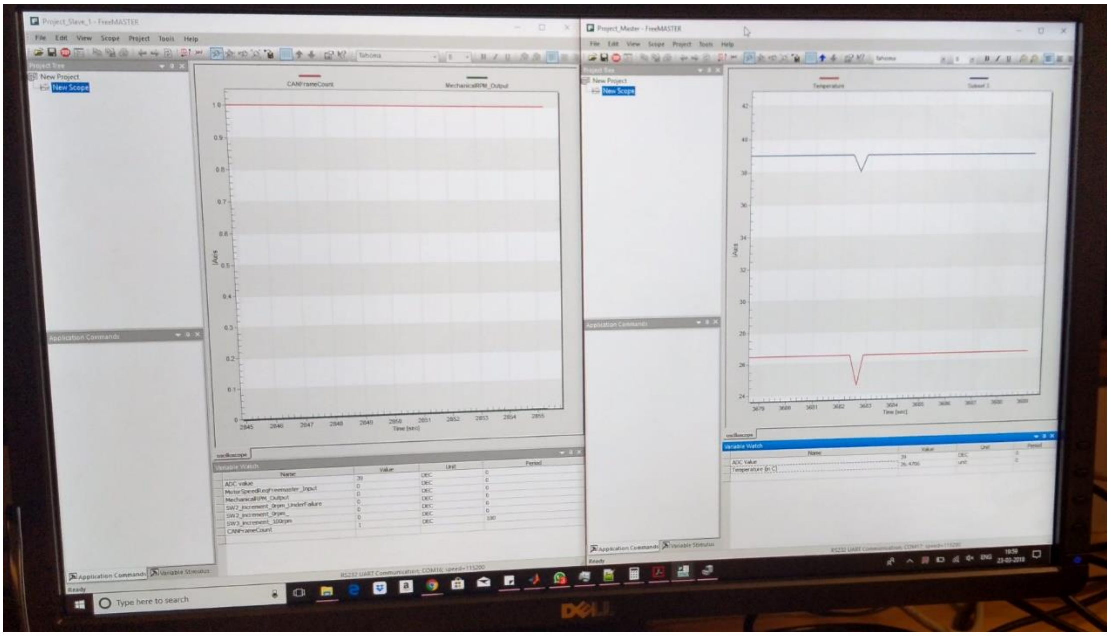
5. Results

The results for the entire operation were obtained on FreeMaster. It is a data visualization software provided by NXP which can be used to view Motor Data pertaining to Speed, RPM, Torque etc. on a real time basis. The test cases can be described with the help of the following diagrammatic representation –



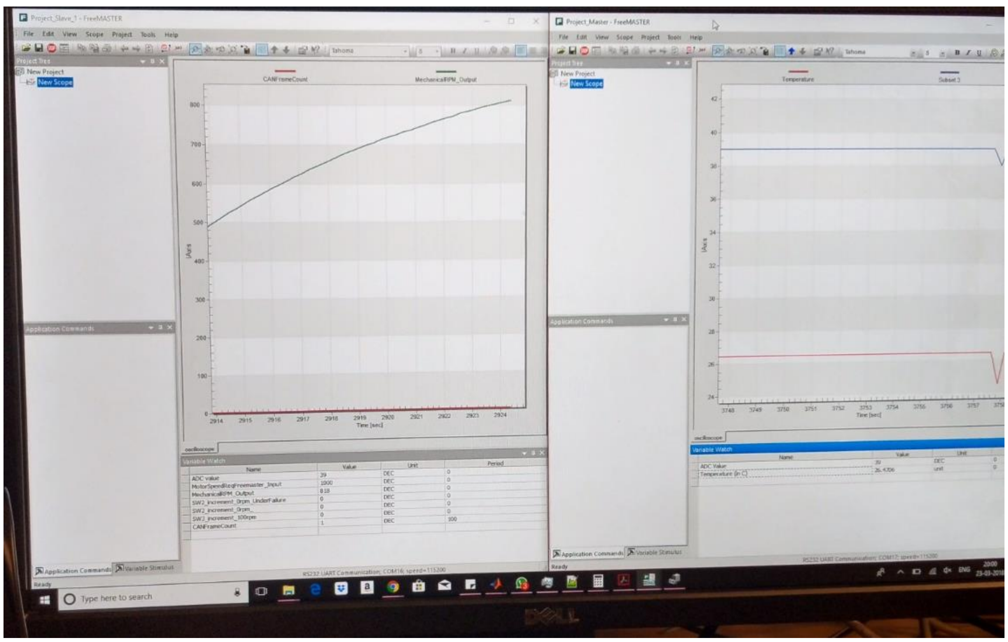
The snapshots for the results were taken based on the different test cases as described below –

1. Initial condition of Motor under Optimal State

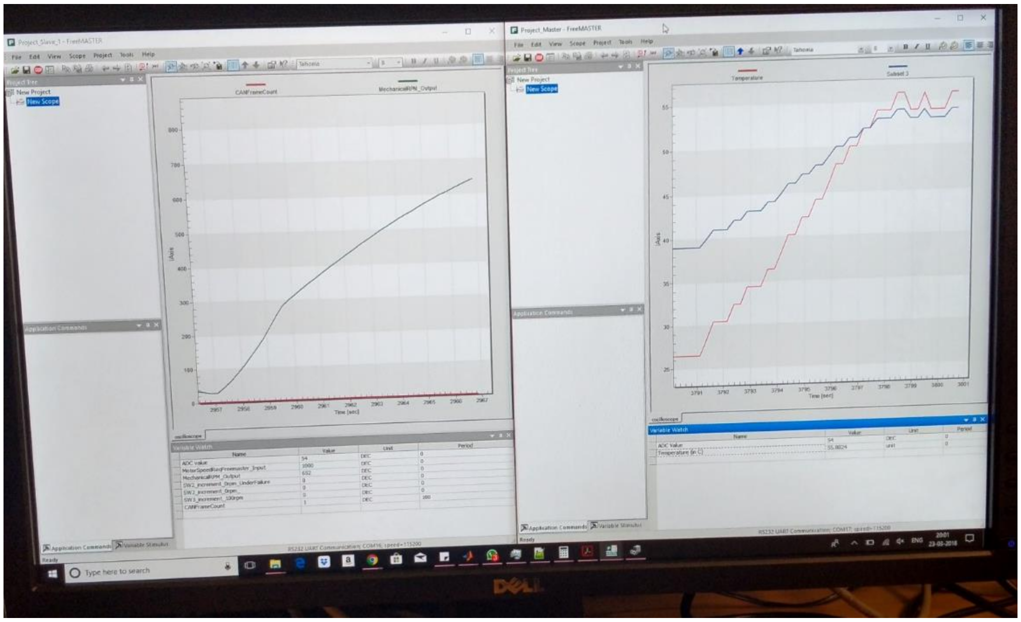


In this condition the Motor is off and the temperature in under optimal conditions.

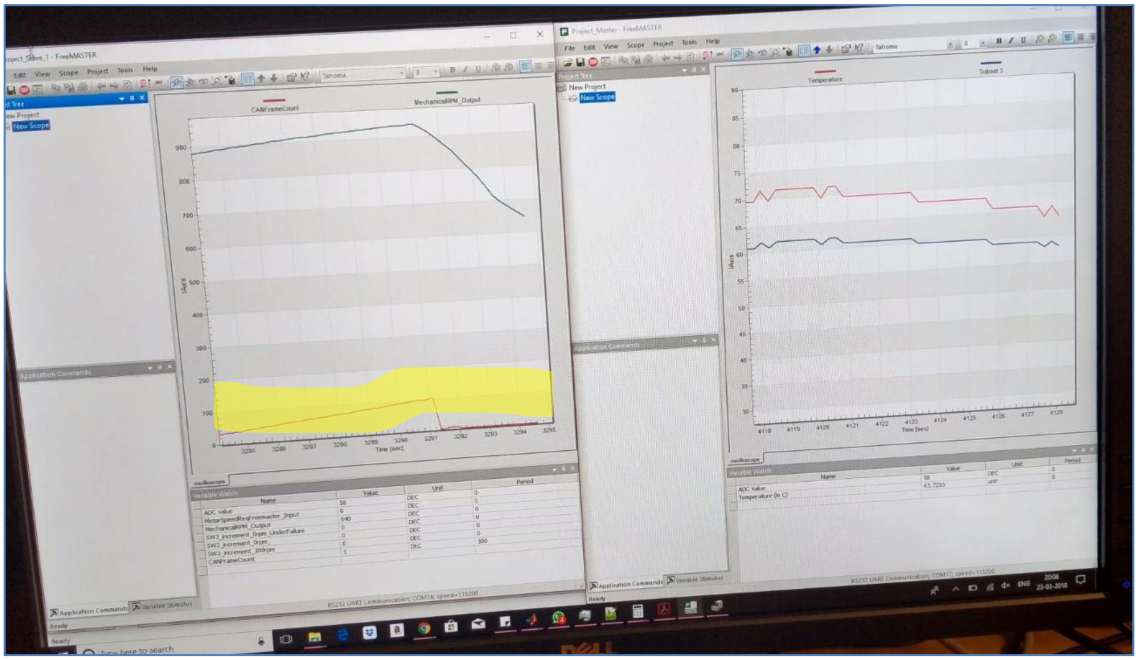
2. Motor RPM to 1000 RPM after pressing SW3



3. Change in temperature due to usage of heat gun



4. CAN Frame Count Reduced after the end of buffer period



6. Advantages and Challenges

- Can be used as a safety application (ISO26262)
- Reduces vehicle component failure thereby making it cost effective.
- Increased fault tolerance.
- The TMP-36 sensor has accuracy $\pm 2\,^{o}$ C. Using a different sensor, we can get improved accuracy

7. Future Scope

- Connection of up to 8 sensors having 64 bits of data thereby making complete utilization of CAN data frame.
- Can use HMI display for real time temperature viewing.

8. Conclusion

Cost effective and ISO 26262 compatible temperature monitoring prototype using CAN for engine safety check has been achieved.

9. Reference

- S32K144-NXP. Retrieved March,2018, from: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/s32-automotive-platform/s32k144-evaluation-board:S32K144EVB
- TMP-36 Analog Temperature sensor. Retrieved March 2018 from: http://www.datasheet4u.com/datasheet-pdf/AnalogDevices/TMP36/pdf.php?id=315262
- LINIX 45ZWN24-40 Motor. Retrieved March 2018 from: http://www.farnell.com/datasheets/2000271.pdf