GANGES – SOLAR CLEANER
FIRMWARE DESIGN DOCUMENT
(RTI-EM-DOC1100-FDD-RV01)

Revision History

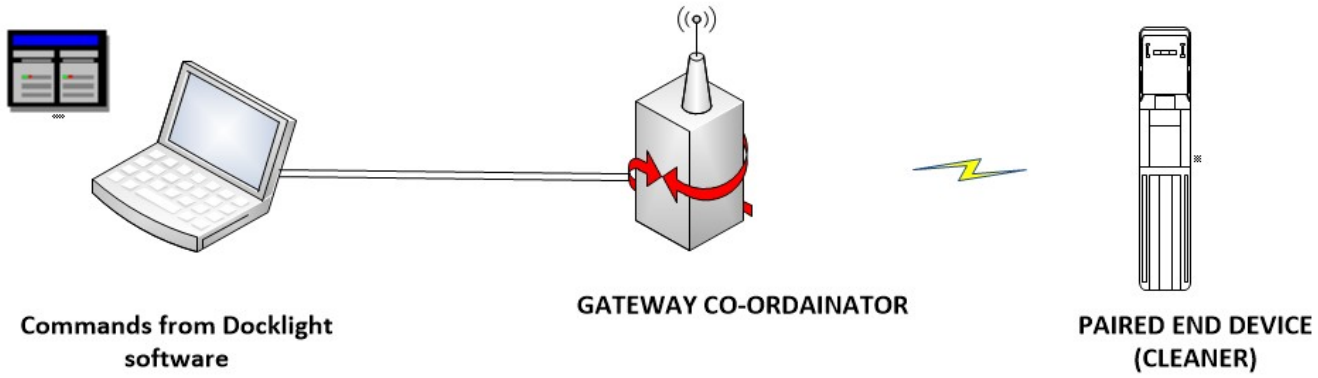
Rev	Date	Change Description	Prepared By	Verified By	Approved By
00	28.05.2022		Shathik, Dinesh		
01	27.07.2022	Change in error code	Dinesh		

Table of Contents

1 .INTRODUCTION

Ganges machine consists of two main units, Solar Cleaner and Track Changer. Solar Cleaner is Cleaning Robot which will work based on two communication methods, Zigbee communication for wireless access and UART Communication for wired access with the Solar Cleaner main card for configuration and monitoring purpose. Track Changer is a linear moving trolley mechanism which acts as docking station for Solar Cleaner which will moves the Cleaner from one row to another row of solar modules.

2 .SYSTEM ARCHITECTURE

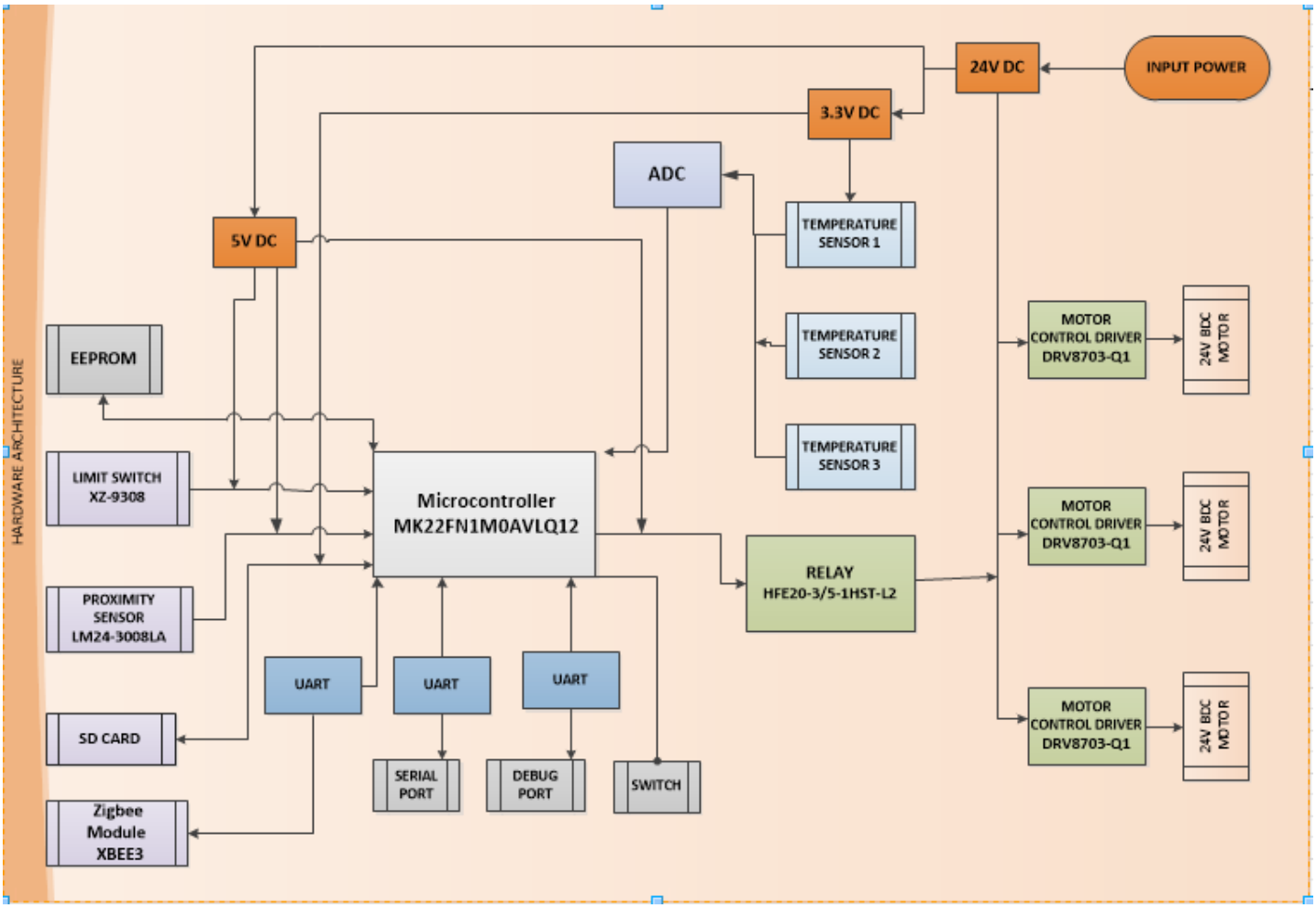


2.1 .SYSTEM DESCRIPTION

The Solar Cleaning Robot works under the Zigbee Communication protocol. The API Command is sent through the Zigbee module from the software along with three parameters such as device id, command and values. The parameters are received to the solar cleaner through the Zigbee connected to the UART communication.

Through the API command we can control the working of the solar cleaner and we can get the sensor values and stored log from the device.

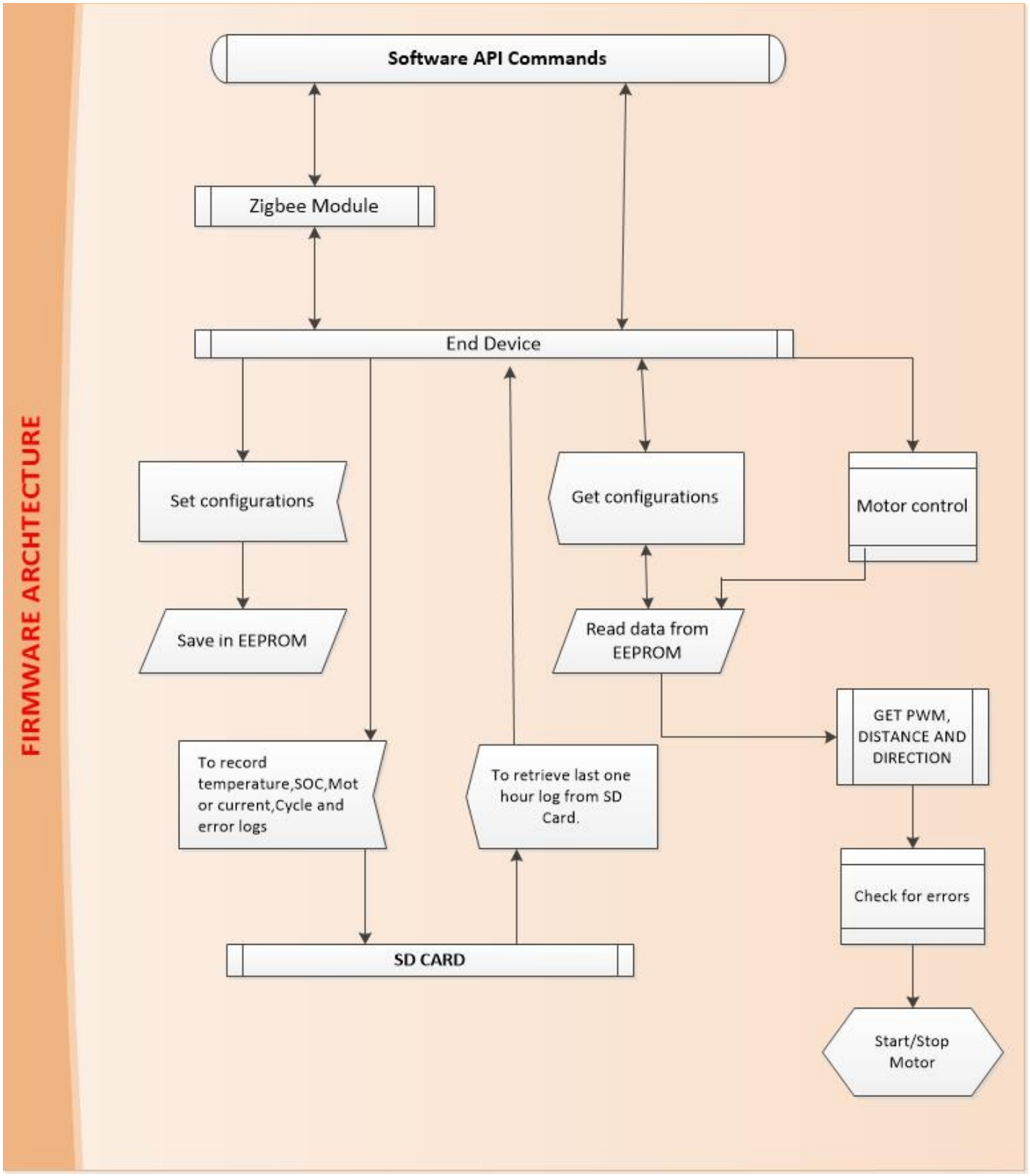
2.2 .HARDWARE BLOCK DIAGRAM



2.3 .HARDWARE DESCRIPTION

Name	Hardware Used	Function
NXP Micro-Controller	MK22FN1M0AVLQ12	
Temperature Sensor	TMP75AIDR	Two IC's is placed on the board which Check Temperature of Motor Section & Battery charge Section One IC is used for Internal board temp
Switch	-NA-	This is to change from local and remote, USB & ZIGBEE select – GPIO is connected with a link switch to ON/OFF
SD CARD	-NA-	For DATA storing, external memory card is added in the PCB
RTC	-NA-	To run real time clock
Rotational Count Sensor	LM24-3008LA (2 proximity sensors)	Two proximity sensors are connected as digital inputs and their status change is recorded here which is used or counting the brush rotation count and Linear movement count
Relay	HFE20-3/5-1HST-L2	To power ON the Relay for Motor Output, to control the relay located in the PCB
Motor Control Driver	DRV8703-Q1	Motor Direction, Motor ON/OFF is controlled through motor driver
Synchronous buck controller/charger	LTC4015EUHFPBF	BMS unit for charging the battery using PV panels and provides regulated power to the cleaner.
LED	17-21SURC/S530-A2/TR8	To show the board status, Communication status
I2C	-NA-	Communication for serial interface, clock, board temperature
EEPROM	-NA-	Storing event log data of Existing
Edge Sensor	XZ-9308 (Limit switch)	Two limit switches are connected in Normally Closed condition, if they change to Normally Open the movement of linear motor is stopped based on the respective digital inputs
Debug	-NA-	Enabling serial port for error debugging
Communication	XB3-24Z8UM	ZigBee Communication is used for wireless communication between Cleaner and System
ADC	-NA-	ADC is used for getting Motor Currents, Battery section voltage references

3 .FIRMWARE ARCHITECTURE


FIRMWARE ARCHTECTURE
Software API Commands
Zigbee Module
End Device
Set configurations
Get configurations
Motor control
Save in EEPROM
Read data from EEPROM
GET PWM, DISTANCE AND DIRECTION
To record temperature,SOC,Motor current,Cycle and error logs
To retrieve last one hour log from SD Card.
Check for errors
SD CARD
Start/Stop Motor

3.1 .FIRMWARE DESCRIPTION

When the cleaner is turned on, the cleaner checks the battery SOC, If Battery SOC is above the minimum SOC set value, then cleaner starts operates from docking station to reverse station and comes back to docking station, during this movement it cleans the PV panels. The operation and features of the machine are described below.

ADC:

The three motors 1 brush motor and 2 linear motor current values are calculated with ADC. The motor faults are set in amps. The set values with command “000F” are compared with these measured adc values . If the current value is above the set value then it is considered as a motor fault condition.

```
void CheckADC (void)
{
  uint64_t ADC0AvgValue;
  uint64_t ADC1AvgValue;

  ADC0AvgValue = 0;
  ADC1AvgValue = 0;
  for (uint16_t i=0;i < ADC_MAX_SAMPLES; i++)
  {
    adc0ConversionDoneFlag = false;
    ADCChannelStart(IMOT1);
    while(!adc0ConversionDoneFlag);
    ADC0AvgValue += ADC0SampleValue;

    adc1ConversionDoneFlag = false;
    ADCChannelStart(IMOT2);
    while(!adc1ConversionDoneFlag);
    ADC1AvgValue += ADC1SampleValue;
  }
```

API processing:

API commands are given to set maximum and minimum values for all the parameters. API commands SET/GET are used to Set the values, to check the stored values, logs of different parameters with respect to date and time, also used for monitoring and control the working of the cleaner.

```
#include "RotateSense2.h"
#include "EdgeSense1.h"
#include "EdgeSense2.h"
#include "SDFAT/SDcardoperation.h"

#define MAX_JASON_VALUE_LENGTH             200U
#define LOG_STR_LEN                        35U
#define MAX_NO_OF_EVENT_LOGS               50

uint64_t receivedMACAddr = 0;

#define CMD_SET_MODE                          0x0001
#define CMD_SET_MOTOR_DIR_MANUAL              0x0003
#define CMD_SET_MOTOR_PWM_MANUAL              0x0004
#define CMD_SET_BRUSH_MOTOR_STATE             0x0005
#define CMD_SET_BRUSH_MOTOR_PWM_MANUAL        0x0006
#define CMD_SET_BRUSH_MOTOR_DIR               0x0007
```

FILE: APIProcessing.c

Communication:

1.Config Uart:

Config UART is used for direct communication with the device. The commands given through dock-light are received and values are transmitted directly through the config UART port.

2. Zigbee:

Zigbee is used for wireless communication with the device. The devices are connected as nodes.

Device ID is given for identification of the device as which device to be communicated.

3. Debug Uart:

Debug Uart is used for microcontroller troubleshooting.

```
ErrorStatus UartTransmit (uint8_t * txBuffer, uint16_t txBufferLength)
{
  ErrorStatus status = ERROR;
  if(GetUartType() == ZIGBEE_UART)
  {
#ifdef ZIGBEE_JSON_MODE
    ConvertHexAndProcess(txBuffer, &txBufferLength);
#endif
    status =ZigbeeApiTxData(GetReceivedMACAddress(),txBuffer, txBufferLength);
  }
  else if(GetUartType() == DEBUG_UART)
  {
    status = DebugUartTransmit(txBuffer, txBufferLength);
  }
  else if(GetUartType() == CONFIG_UART)
  {
    ConvertHexAndProcess(txBuffer, &txBufferLength);
    status = ConfigUartTransmit(txBuffer, txBufferLength);
  }
  return status;
}
```

Default values:

The default values are applied whenever the PCB is cleared and downloaded the program or when we execute the default value command from docklight. These default values are configured and updated through API commands from the software.

```
void SetDefaultDevIDInfo(void)
{
  stDevInfo *defDevInfo = GetSetDeviceIDInfo();
  snprintf(defDevInfo->devID, MAX_DEV_INFO_FIELD_LEN, DEFAULT_DEV_ID);
  snprintf(defDevInfo->hwVersion, MAX_DEV_INFO_FIELD_LEN, DEFAULT_DEV_HW_REV);
  snprintf(defDevInfo->serialNo, MAX_DEV_INFO_FIELD_LEN, DEFAULT_DEV_SERIAL_NO);
}

void SetDefaultPwmCycleManualModeParameter (void)
{
  stRobotPwmParam *robotPwmParam = GetSetCycleManualPwmParameters();
  GetDefaultMotionParameter(robotPwmParam);
  robotPwmParam->steadyPwm1 = DEFAULT_MANUAL_PWM_DUTY;
  robotPwmParam->steadyPwm2 = DEFAULT_MANUAL_PWM_DUTY;
}
```

Delay:

Delay pauses the program for the amount of time (in milliseconds) specified as parameter.

```
void DelayMsTimerhandler(void)
{
  delayMillisecondTime++;
}

uint64_t GetDelayMsTimeValue(void)
{
  return delayMillisecondTime;
}
```

Edge Sensor:

Two limit switches are connected in normally closed, if state is change to normally open the movement is stopped in that direction and the cleaner will not move from there. The status of the sensors will be shown as 0 and 1 i.e. engaged and disengaged.

Command to view Edge Sensor states: 100E

```
bool EdgeSenseInit (void)
{
  bool status = true;
  status &= EdgeSense1Init();
  status &= EdgeSense2Init();
  return status;
}

bool *GetSetEdgeSensorEnabledState(void)
{
  return(&edgeSensorEnabledState);
}
```

FILE: EdgeSenseCommon.c

EEPROM:

Configuration values are stored in EEPROM. Values like motor PWM, auto schedule time, wheel dia are stored in EEPROM. The stored values are retrieved and processed during the motor operations.

```
eEepromStatus WriteEEPROM (uint16_t storeAddress, float *storeData, uint16_t noOfData)
{
  union {
    float storeFloat;
    uint16_t storeInt[2];
    uint32_t storeInt32;
  } unStoreData;

  eEepromStatus status = EEPROM_OK;
  /* Unlock the Flash Program Erase controller */
  if(FLASH_Unlock() != FLASH_COMPLETE)
  {
    status = EEPROM_UNLOCK_ERROR;
  }
```

I2C:

I2C Communication is used with the temperature sensor to communicate and read the values from sensor. Motor current and battery current values are transmitted in I2C. The read values are calculated with ADC.

```
void TempSenseTimer_ms(void)
{
TempSensorI2cTimer++;
}

uint32_t GetTempSenseI2CTimer(void)
{
return TempSensorI2cTimer;
}

void InitI2C(void)
{
  uint32_t sourceClock = 0;
    BOARD_I2C_ReleaseBus();
    I2C0_InitPins();
```

LTC4015:

LTC4015 is used as battery charge controller to charge Lithium phosphate battery. PV voltage is also monitored & recorded. This is the primary source for PCB power ON even if battery is not connected PCB will be in ON by PV power. Data is transmitted in I2C from LTC4015.

```
LTC4015 chip;
port_configuration_t I2C_Data_2;

enum {
 success = 0,
 failure = !success
};

volatile bool smbalert = 0;

uint8_t batteryError = 0;

static float prevSOCPercent = 0;

uint8_t GetIChargeValue(float current);
```

Motor Control:

1.Brush Motor:

Brush Motor is initialized during the power ON and controlled manually or in auto mode upon set schedule time. The PWM parameters in which motor to be controlled are retrieved from EEPROM data. The PWM parameters are configured through dock-light commands.

```
static void BrushMotorInitPins()
{
    gpio_pin_config_t out_config = {
    kGPIO_DigitalOutput, 0,
  };

  GPIO_PinInit(BOARD_MOTOR_ENABLE_GPIO, BOARD_BRUSHMOTOR_ENABLE_GPIO_PIN, &out_config);
  BrushMotorDisable();
}
```

2. Linear Motor:

Linear Motor is initialized during the power ON and controlled manually or in auto mode upon set schedule time. The PWM parameters in which motor to be controlled are retrieved from EEPROM data. The PWM parameters are configured through dock-light commands.

```
void Motor1Init (void)
{
  Motor1Init_hal ();
}

void Motor1FSM (void)
{
  eMotor1State motor1TargetState = GetMotor1TargetState()
  eMotor1State motor1PresentState = GetMotor1State();

  if(IsLinearEnabled())
  {

  }
  else
  {
    SetMotor1TargetState(MOTOR1_STOP);
    SetMotor1State(MOTOR1_STOP);
  }
```

Port Interrupt:

The situation where the execution of the program is suspended and the processor is forced to take care of the important event is called an interrupt. The interrupt request can be issued by a hardware built into the microcontroller, but can also be issued by an external hardware connected to a pin of a port.

```
void EnablePortInterrupt (void)
{
  EnableIRQ(PORTA_IRQn);
}

void DisablePortInterrupt (void)
{
  DisableIRQ(PORTA_IRQn);
}

void PORTC_IRQHandler(void)
{
  uint32_t volatile interruptFlag
```

Relay:

To power ON the Relay for motor output- to control the relay located in the PCB. To turn on the relay always in manual mode and to turn on the relay in auto mode before operation starts and off the relay after cycle completed.

```
static void RelayOnPulse (void)
{
  GPIO_PortClear(RELAY_GPIO, 1U << RELAY_OFF_GPIO_PIN);
  GPIO_PortSet(RELAY_GPIO, 1U << RELAY_ON_GPIO_PIN);
}

static void RelayOffPulse (void)
{
  GPIO_PortClear(RELAY_GPIO, 1U << RELAY_ON_GPIO_PIN);
  GPIO_PortSet(RELAY_GPIO, 1U << RELAY_OFF_GPIO_PIN);
}

static void RelayNoPulse (void)
{
  GPIO_PortClear(RELAY_GPIO, 1U << RELAY_ON_GPIO_PIN);
  GPIO_PortClear(RELAY_GPIO, 1U << RELAY_OFF_GPIO_PIN);
}
```

Robot Control:

Cleaner robot runs on the pre-set values like ramp up, down time, run pwm, approach start, end pwm, acceleration, deceleration count difference. Distance is converted into pulse count. Soft limit will be included and hard limit switch will be converted as external emergency cut off. To record and log the Motor current with respect to distance and time.

Command to view direction of robot: 100E

```
void RobotFSM (void)
{

//  char getTime_str1[40];
  static float acelStartCount = 0,decelStartCount = 0;
  eRobotState presentRobotState = GetRobotState();
  static bool dirChangeFlag = false;
  static bool normalStopFlag = false;
  switch (presentRobotState)
  {
   case ROBOT_IDLE:
    currentPwm1Duty = IDLE_PWM_DUTY;
    currentPwm2Duty = IDLE_PWM_DUTY;
    StopRotateSenseCount();
 //   prevCount = GetRotateSenseCount();
    SetMotor1AllPwm(currentPwm1Duty);
    SetMotor2AllPwm(currentPwm2Duty);
    break;
   case ROBOT_START:
```

Robot Operation:

In this function the motor operation is controlled i.e, in which direction the motor to be moved, the speed of the motor are controlled here by retrieving the data from EEPROM. Here it checks for auto/manual mode. If manual mode the robot operates on given commands(forward/reverse), if auto mode it checks for scheduled time is equal to present time and cycle starts. During cycle check for any errors and robot needs to be stopped if any error occured. Robot tries to restarts upto set error restart count value (command 0034/1034). If the count value exceeds motor needs abrupt stop and starts after clearing the error.

Command to operate robot manually: 0003

Command to schedule & operate robot in auto mode: 000B

Command to to run maximum number of schedules in auto mode: 0032

```
void RobotOperate (void)
{
  eRobotCommand command = GetMotionCommand();
  stReturn *setReturnValue = GetSetReturnLimits();

  AssignCommand(command);
  if((command != ROBOT_CMD_NONE) && (command != ROBOT_CMD_IDLE)
  {
    //SetLogEvent (EV_LOG_MOT_CMD, (uint8_t)command);
    if((Robot_Error_Count < setReturnValue->Return1))
    {
      ClearFaultsOnRobotCommand();
    }

  }
  //CheckCommunicationError();
  CheckError();
  OperateMotionModes();
  OperateMotionPauseFSM();
}
```

Rotation Sensor:

Two proximity sensors are connected as digital inputs for brush rotation count and Linear movement count. The values are recorded as pulse, calculation is done in logic were pulse is converted into distance in millimeters for linear motor and rotation per minute for brush motor. Wheel diameter is SET by the user in millimeter.

Command to view Rotate Sensor states: 100E

```
#include "RotateSense1.h"
#include "RotateSense2.h"
#include "RobotOperation.h"

#define COUNTSENSOR     ROTATESENSOR_1          // ROTATESENSOR_2

static bool RotateSensorEnabledState = true;

bool RotateSenseInit (void)
{
  bool status = true;
  status &= RotateSense1Init();
  status &= RotateSense2Init();
  return status;
}
```

FILE: RotateSenseCommon.c

RTC:

RTC is present to keep accurate time of day. Time in this device runs as a backup with the charge of capacitor. The auto schedule time is compared with the current RTC time and controls the motor operation if present time of the device and auto schedule time is equal.

Command to view RTC: 100D

```
bool RTCInit (void)
{
  bool status = false;
  rtc_config_t rtcConfig;
  /*
  * rtcConfig.wakeupSelect = false;
  * rtcConfig.updateMode = false;
  * rtcConfig.supervisorAccess = false;
  * rtcConfig.compensationInterval = 0;
  * rtcConfig.compensationTime = 0;
  */
  CLOCK_EnableClock(kCLOCK_Rtc0);

  RTC_SetOscCapLoad(RTC, 0);
```

SD FAT:

SD Card initialization is done if SD Card present in the hardware and mount status is set as true.

If the mount status is true then the log values will be recorded in separate file created for each section(temperature, SOC, motor current, cycle log, error log). Temperature and SOC log are recorded for

every interval set in the command (0030). Motor current is recorded for every 5 seconds when the motor is running. Auto schedule is recorded for every positive and negative direction in a cycle with respect to time and distance. Error log is recorded whenever error is occurred.

```
void SDcardOperation(void)
{
  if(SD_MOUNT_STATUS == 1)
  {
     Temperaturelog_main();
     SOClog_main();
     MOTORCRNTlog_main();
     ERRORlog_main();
     AutoCycle_main();
  }
}
```

Command to view log from SD Card: 103F

SPI:

SPI communication is used for pwm and SD Card read/ write. The controller acts as a master and SD Card as slave. Whenever need to write a data in SD Card the chip select pins are made high and data is transferred through SPI.

```
void DSPI_MasterUserCallback(SPI_Type *base, dspi_master_handle_t *handle, status_t status, void *userData)
{
    if (status == kStatus_Success)
    {
        __NOP();
    }

    isTransferCompleted = true;
}
```

Temperature Sensor:

Temperature Sensors are used to check temperature of motor section, battery charge section and internal board temperature, values are recorded with respect to date and time, values will be recorded on specific interval which is configurable time between 1 to 60 minute. For reading we will get as per the request between start time and end time, recorded temperature values are stored in SD Card.

```
float GetTemperatureSensorData(eTempSensors sensorId)
{
  uint8_t slaveAddress;
  uint32_t currTime = GetTempSenseI2CTimer();
  switch(sensorId)
  {
  case TEMPSENSOR_1: if((currTime - tempSense1_ptime) >= TEMP_SENSOR_READ_AGAIN)
  {
    slaveAddress = TEMP_SENSOR_1_ADDRESS;
    tempSense1_ptime = GetTempSenseI2CTimer();
    break;
  }
  else
  {
    return tempSense1_pvalue;
  }
```

FILE: temp_sensors.c

Command to view Temperature Sensor Readings: 100D

Switch:

To switch between manual and auto control in communication. In manual control, API Commands are directly given to the controller through configuart. In auto control device is paired with ZigBee and commands are given through ZigBee and communicated as wireless communication.

```
eControlModes GetSwitchState(void)
{
  switchState = (GPIO_PinRead(SWITCH_GPIO, SWITCH_PIN) == SWITCH_VALID_STATE);
  if(!switchState)
  {
    return MANUAL_CONTROL;
  }
  else
  {
    return AUTO_CONTROL;
  }
}
```

4 .FIRMWARE FEATURES

- ZigBee XB3-24Z8UM communication is used for wireless communication with user and solar cleaner.
- SD CARD for storing temperature, motor current, error log, cycle log and SOC with respect to date and time.
- Defined APIs for hardware control and monitoring.

5 .FIRMWARE FLOW

5.1 .FIRMWARE FLOW- PROCESS-1

Initial Operation when PCB Powered ON.

1. Default Welcome Message:

******REN CONTROLLER – D1.6V******

- This welcome message is displayed in dock light when the PCB is powered ON.

2. Initializations:

- RTC is initialized – if RTC is valid no need to set RTC. If invalid RTC error(100) is created and need to set RTC (000D) and clear the error(0010).
- ADC Initialization – ADC is initialized to get the motor current.

- EEPROM Initialization – EEPROM is initialized and the stored values are initialized in the respective variables.
- Relay Initialization, Robot Initialization – Relay and 2 Linear Motors and 1 Brush Motor are initialized.
- Rotate Sensor and Edge Sensor Init – Rotate sensor for pulse count, Edge sensor for robot stop.
- I2C, SPI Initialization – I2C is initialized for communication with ADC, SPI for communication with SD Card.
- ZigBee UART, Con-fig UART Init – ZigBee UART is initialized for communication with ZigBee. Con-fig UART is initialized for communication through serial port.
- Update Motor Offset Value – Initial motor readings are updated for calculations.

3. LED:

- When the device is initially powered ON, the communication LEDS LED1 initially blinks for 3 seconds and LED2 and LED3 blinks for 1 sec and turned OFF. Then LED1 blinks for every 3 seconds. LED14 will be turned ON and OFF once the PCB is powered ON and then it continuously blinks for 2 times for every 1 second.
- If the communication is established through serial port and switch is connected for communication then LED3 blinks for every 2secs with parallel to LED1. If any command is sent through serial port then LED3 blinks for 0.3 sec.
- If the communication is established with ZigBee and if any command is received through ZigBee UART, LED13 blinks for 5 sec and turned OFF. The LED14 turns ON and OFF 2 times for every 1 sec.
- After power ON if cleaner is in Manual mode LED7 is powered ON, if cleaner is in AUTO mode LED7 is OFF.

5.2 .FIRMWARE FLOW- PROCESS-2

1. API commands processing:

- API commands are received to the PCB through dock-light software via con-fig UART or ZigBee UART.
- For transferring the commands through the con-fig UART, connect through serial port of the PCB and switch must be enabled.
- For transferring commands through ZigBee the PCB must be in communication with the gateway card and cleaner PCB acts as a co-ordinator.
- In case of any invalid commands or data insufficiency or data not in range or command invalid. Error is given and not processed.

2. Robot Operation:

- Robot operates on two modes. 1) Auto 2) Manual. *(Command: 0001)*
- In manual mode forward/ reverse/ cycle command is give. *(Command: 0003)*
- In forward command, robot starts from the home position and cleans upto the given distance and stops at target distance or if limit switch engages in-between robot stops the operation. *(Command: 0003)*
- In reverse command, robot starts from end position and moves to home position and stops at set target distance or if reverse limit switch is engaged it stops the operation. *(Command: 0003)*
- With cycle command, robot starts moving from the home position and runs till the set forward target distance. After reaching the forward target distance, robot starts moving reverse and comes to home position till reverse target count. *(Command: 0003)*
- In auto mode, total of 19 schedules and 19 cycle frequency can be set. Robot works on total number of cycle frequency per day. If the total number of schedule set is 15 and total number of cycle frequency set is 5. Robot operates 5 schedule cycles per day.

Command to switch between modes: 0001

Command to set 19 auto-schedule time in hours: 000B

Command to set cycle frequency per day: 0032

3. Robot FSM:

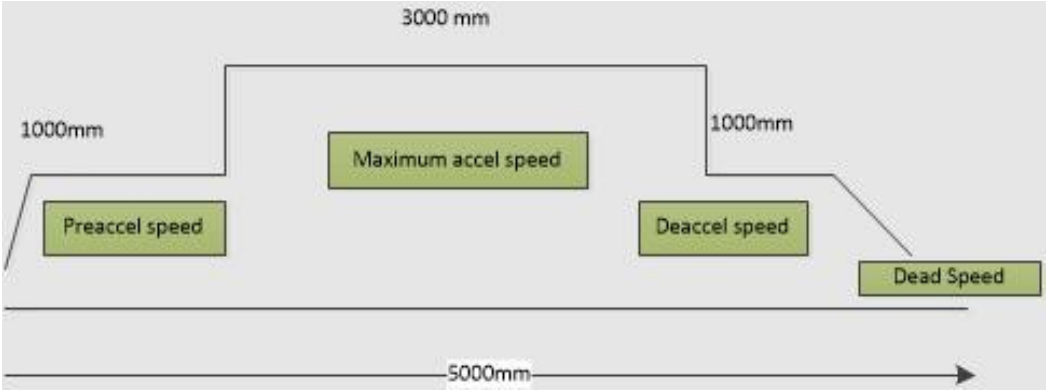
- Initially when the robot starts its operation in auto or manual mode. The robot starts run in preaccel speed upto the set distance(1000mm) and after reaching it robot starts operating in its maximum speed(3000mm) and after completing the distance upto maximum speed robot slows down its speed and starts running in deaccel speed(1000mm) upto the target distance.

Command to set pwm values for linear motor in manual & auto mode: 0004 & 0009.

Command to set pwm values for brush motor in manual & auto mode: 0006 & 000A.

- After deaccel state robot runs in a dead speed to cover the total distance set by the user. Since the target distance cannot be achieved with pulse count. Calculations are done and remaining distance are covered as dead speed with timer.

Set dead speed value for motor: 0031



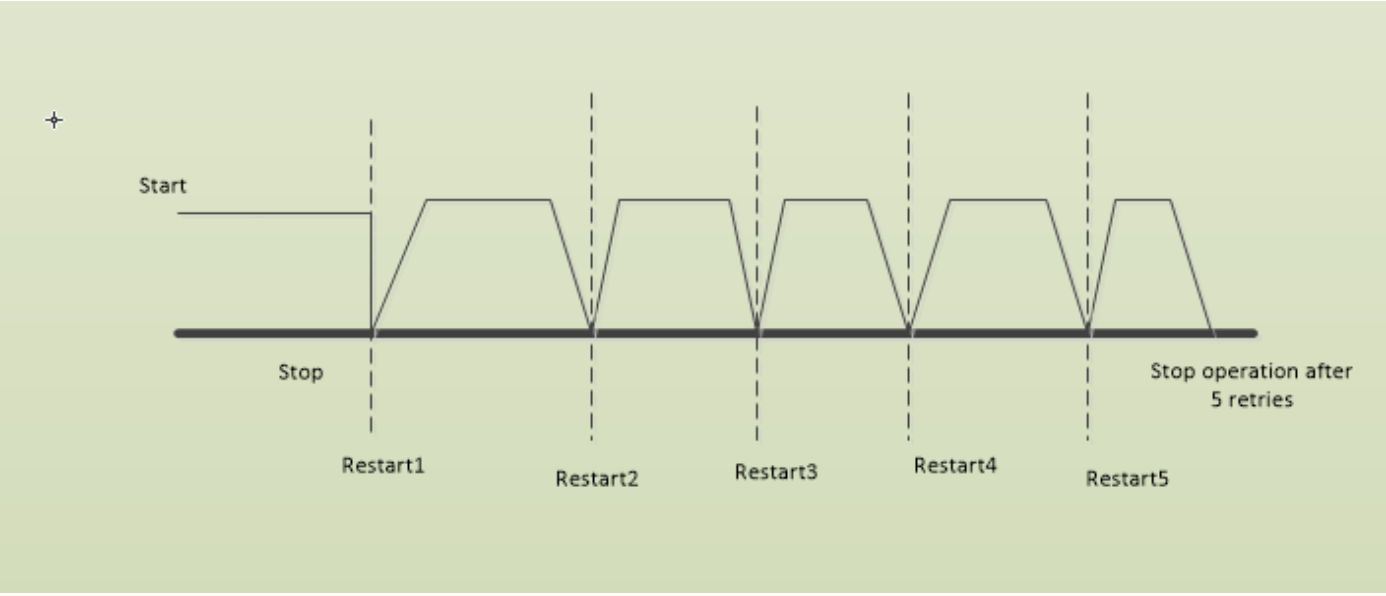
5.3 FIRMWARE FLOW- PROCESS-3

1. Error Operations:

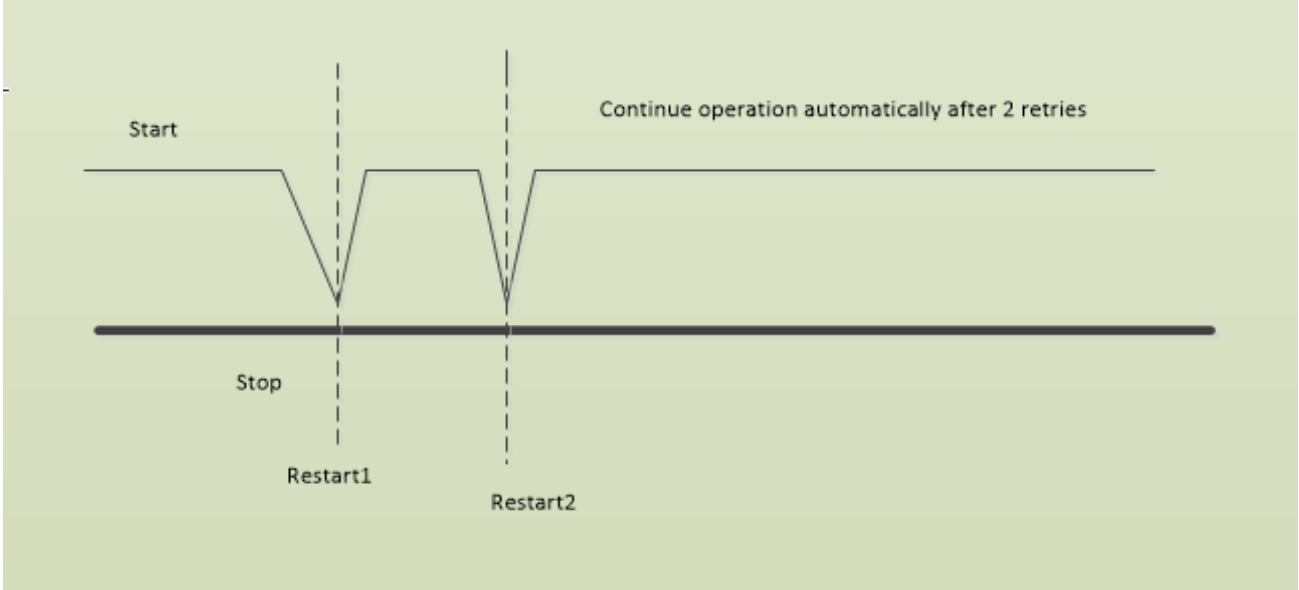
During robot operation robot checks for some errors like Motor over current fault, minimum battery voltage fault, overboard temperature fault, battery fault, communication fault, ZigBee fault and RTC fault. These faults are created to avoid operation of motor in unsafe conditions.

If an error is created when robot is running the robot stops its operation and checks for auto restart count set in command 0034. If the restart count is 5, robot retries for 5 attempts.

For example if motor overload error is created:



Robot stops its operation and checks whether the error still repeats and restarts its operation. This process is done upto set number of restart counts. Then the user needs to clear the error and continue the operation manually.



If set number of restart count is 5 and error is cleared automatically within 2 retries then the cycle continues normally upto the target distance.

Command to set maximum error restart counts: 0034. (Restart count can be set upto 10)

Cleaner Errors:

Motor over current error(2,4,8):

when the cleaner is operating if the errors got generated, set the over current limits in 000F command.

Minimum battery voltage fault(10):

set the minimum battery SOC value in command 0012.

Over board temperature fault(20):

Error occurs when board temperature is higher.

Battery fault(40):

Error occurs when Battery is not working properly.

ZigBee fault(80):

Error occurs when ZigBee is not working properly.

RTC fault(100):

Error occurs when RTC is not working properly, RTC value not updated set RTC value in command 000D.

2. File creation and storing logs:

Files are created(Temperature log, SOC log, Motor current log, Cycle log, Error Log) in SD card initially during power ON.

Write:

Temperature and SOC logs are stored for every interval set in 0030 command. If the interval is 1 minute for every 1 minute temperature and SOC logs are written in the SD Card with current time.

Motor current is stored in SD Card for every 5 seconds during motor operation with respective time. When motor is idle no data is written in SD Card.

Cycle log is recorded in SD Card with respective start time, end time, distance travelled between cycle start and end.

Error log is recorded once during PCB power ON and whenever other errors are created during motor operation.

Command to set time interval for storing data in SD Card: 0030

Read:

From SD Card, last 1 hour data can be read with command 103F with values 1 - for temperature log, 2 – for SOC log, 3 – for motor current log, 4 – for cycle log, 5 – for error log, 6 – to delete all logs and create new file.

Command to get logs from SD Card: 103F

Maximum logs viewed in dock-light through ZigBee and serial port:

S.NO	LOG	MAX LOG AVAILABLE THROUGH ZIGBEE	MAX BYTES	MAX LOG AVAILABLE THROUGH SERIAL PORT
1	TEMP	37 logs	1830	60 logs
2	SOC	27 logs	1400	60 logs
3	MOTOR CURR	30 logs	1350	300 logs
4	CYCLE LOG	16 logs	1440	30 logs
5	ERROR LOG	34 logs	1440	60 logs

API Commands:

S.NO	OPERATION	COMMAND	DIRECTION
1	Set Mode –Auto/Manual	0001	H —> D
2	Set Manual Control	0003	H —> D
3	Set motor PWM Value - Manual Mode	0004	H —> D
4	Set Motor Status	0005	H —> D
5	Set Brush motor PWM Value - Manual Mode	0006	H —> D
6	Set Brush Direction	0007	H —> D
7	Set Motor PWM value — Auto mode	0009	H —> D
8	Set Brush motor PWM value — Auto mode	000A	H —> D
9	Set Auto Schedule Time of Day	000B	H —> D
10	Set Sensor Status	000C	H —> D
11	Set RTCC Value	000D	H —> D
12	Set Device Info	000E	H —> D
13	Set Motor Over Current limits	000F	H —> D
14	Clear Error Code	0010	H —> D
15	Get Battery, PV, SOC values	101C	H <--> D
16	Get Error Status	1018	H <—> D
17	Get Device Info	1019	H <—> D
18	Get Current Mode	1001	H <—> D
19	Get Manual Control	1003	H <—> D
20	Get motor PWM Value — Manual Mode	1004	H <—> D
21	Get Motor Status	1005	H <—> D
22	Get Brush motor PWM Value — Manual Mode	1006	H <—> D
23	Get Brush Direction	1007	H <—> D
24	Get Motor PWM value — Auto mode	1009	H <—> D
25	Get Brush motor PWM value — Auto mode	100A	H <—> D
26	Get Auto Schedule Time of Day	100B	H <—> D
27	Get RTCC Value	100D	H <—> D
28	Get Sensor Status	100C	H <—> D
29	Get Current Status	100E	H <—> D
30	Get Last Operation Status	100F	H <—> D

31	Get Motor Over Current Limits	1010	H <—> D
32	Get Motor Current Values	1011	H <—> D
33	Set Rotational Pulse Count values	0011	H —> D
34	Set Low Battery SoC value	0012	H —> D
35	Get Rotational Pulse Count values	1014	H <-> D
36	Get Low battery SoC value	1015	H <-> D
37	Set Zigbee Configuration	0013	H --> D
38	Get Zigbee Configuration	101A	H <-->D
39	Get Zigbee Network Parameters	101B	H <--> D
40	Reset Zigbee Network	0014	H --> D
41	Get Temperature Values	101D	H <--> D
42	Set Over Current Fault Conditions	0016	H --> D
43	Get Over Current Fault Conditions	101E	H <--> D
44	Reset to Defaults	0015	H --> D
45	Gateway Zigbee Configuration	0201	H --> D
46	Get Gateway Zigbee Configuration	1201	H <-->D
47	Get Gateway Zigbee Network Parameters	1202	H <--> D
48	Reset Gateway Zigbee Network	0202	H --> D
49	Set / Get Interval for Different parameters	0030 / 1030	H —> D / H <--> D
50	Set / Get Wheel diameter in Millimeter	0031 / 1031	H —> D / H <--> D
51	Set / Get no of cycle frequency per day	0032 / 1032	H —> D / H <--> D
52	Set / Get Home Return / No operation States	0033 / 1033	H —> D / H <--> D
53	Set / Get Error Reset Count	0034/1034	H —> D / H <--> D
54	Get Log for Specific parameter	103F	H <--> D