ICNTINMST-2021 ISSN: 2008-8019

Special Issue on Proceedings of International Conference on Newer Trends and Innovation in Nanotechnology, Materials Science, Science and Technology March 2021. International Journal of Aquatic Science, Vol 12, Issue

RTO's technology-based parking system

Mohammed Raffic, N[1*] , P. Siva[2], and Balthilak, A[3]

[1,2&3] Department of Mechanical Engineering, Nehru Institute of Technology, Kaliyapuram, Coimbatore 6411 05, Tamilnadu, India

*Correspondent Email: wasimraffic@gmail.com

Abstract: The main goal of this project is to create a system that takes control of the car and automatically opens and closes doors without the assistance of the user. The system can be fully controlled from a Smartphone within a 10-meter range using a special app. Unlike some previous generation systems, the user does not need to open or close the door. Instead, a user standing outside with his or her smartphone can now fully control the park and retrieve system. Bluetooth is used to communicate between the Android Smartphone and the car. The car unit will be connected to the Smartphone via Bluetooth, removing the majority of the security concerns. This project also aims to create a self-driving car that can be started and driven using an Android smart phone. This will provide the user with additional functionality, especially when retrieving. Controlling the door from a distance is possible.

***Keywords*—smart phone, retrieve system, blueetooth, security, distance**

I. INTRODUCTION

The number of vehicles on road is on the rise throughout the world and as a result the total amount of traffic is increasing rapidly and hence the parking spaces are getting fewer and smaller day by day. The car driver finds very little space to correct a vehicle maneuver while parking and the hectic nature of today's driving habits makes the situation much worse. Hence Automatic Park Assist technology was developed which helps a car to steer itself into a parking space with little input from the user. High end cars already have this feature. But in the existing system the user needs to select auto parking mode by means of pushing a button on the dashboard and the system scans the parking area and drives the vehicle to a free space.

Everyone knows that parking is only one half of the scenario. The other aspect is taking it out. To retrieve a parked car, the existing systems still doesn't have an automated solution. It still relies on the driver to get in. Hence to make it even we came up with an innovative idea that will automatically handle the parking as well as the retrieving process of a car as shown in fig.1. This feature is a powerful solution for drivers who are beginners in driving and for those who meet with a parking space that are too tight or a very narrow garage. Instead of simply pushing a button on the car dashboard and sit silently inside an automated car that park by itself the user can get out of the car and ask it to park by using an android application from his/her Smartphone. In retrieve mode the user can automatically retrieve the car from the parking space to the place where the driver initially got alighted.

Fig. 1 Retrieve assist system

II . AUTONOMOUS SELFPARKING

Automatic parking or self parking for an autonomous car maneuvering system is achieved by using the special app on the android phone called BT interface. This app can be downloaded from the play store of the Smartphone. The special app is installed after the download and can be used to control the car unit. Initially the Smartphone is paired with the car unit through Bluetooth connectivity and then park mode is opted for automatic parking which makes the vehicle to move from a traffic lane into a parking spot to perform parallel parking, perpendicular or angle parking. The automatic parking system aims to enhance the comfort and safety of driving in constrained environments where much attention and experience is required to steer the car. The parking maneuver is achieved by means of coordinated control of the steering angle and speed which takes into account the actual situation in the environment to ensure collision-free motion within the available space. An automatic parking system uses various methods to detect objects around the vehicle. Sensors installed on the front and rear bumpers can act as both a transmitter and a receiver. These sensors send a signal that will be reflected back when it encounters an obstacle near the vehicle. Then, the carputer will use the time signal it receives to determine the position of the obstacle. Other systems mounted on the bumper use the camera or radar to detect obstacles. But the result is the same: the car will detect the parking space size and distance from the roadside, then drive the car into the parking space. The controller diagram in the vehicle section is shown in fig.1.

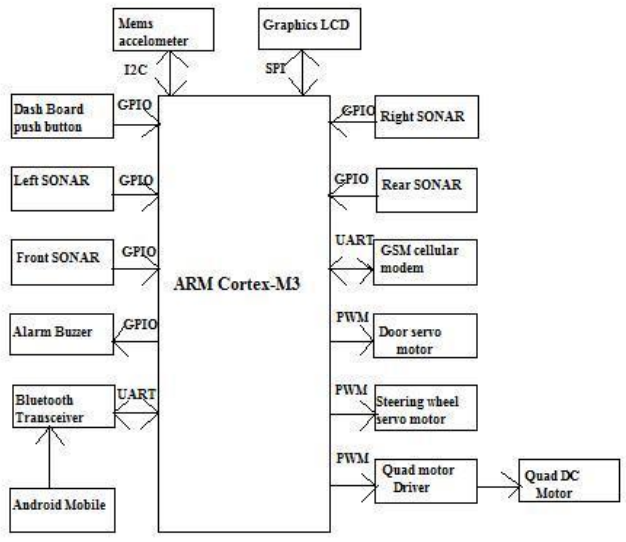


Fig.1 Robotic vehicle unit

A. CODING

The C coding for the controller to perform the requested parking operation based on RTOS technology

```
#include "LPC17xx.h"
#include "config.h"
#include "gpio.h"
#include "timer0_delay.h"
#include "timer1_int.h"
#include "pwm.h"

#include "board.h"
#include "glcd_5110.h"
#include "uart.h"
#include "24lc02.h"
#include "lis302dl.h"
#include "sonar_sr04.h"
#include "wheel.h"
#include "gsm_sim300.h"

/* Scheduler includes */
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"

/* Standard includes */
#include
<C:/nxp/LPCXpresso_6.1.4_194/lpcxpress
o/Tools/arm-none-
eabi/include/redlib/string.h>
#include
<C:/nxp/LPCXpresso_6.1.4_194/lpcxpress
o/Tools/arm-none-
eabi/include/cr_section_macros.h>

#if CONSOLE_PRINT
#include
<C:/nxp/LPCXpresso_6.1.4_194/lpcxpress
o/Tools/arm-none-
eabi/include/redlib/stdio.h>
#endif

/*-----------------------------------------------
*/

#define ROBOT_CHECK_ENABLE0

/*-----------------------------------------------
*/

/* Global */
xQueueHandle xBtiCmdQueue = NULL;
xSemaphoreHandle
xRemoteAlertTaskBeginSemaphore        =
NULL;
xSemaphoreHandle
xRemoteAlertTaskEndSemaphore          =
NULL;

/*-----------------------------------------------
*/

typedef enum _BTI_COMMAND {
        BTI_CONNECTED,
        FORWARD,
        REVERSE,
        RIGHT,
        LEFT,
        STOP,
        PARK_RIGHT,
        PARK_LEFT,
        RETRIEVE,
        SMART_DRIVE,
        DOOR_OPEN,
        DOOR_CLOSE
} BTI_COMMAND;

typedef struct _SONAR {
	uint16_t front;
	uint16_t right;
	uint16_t left;
	uint16_t rear;
} SONAR_TYPE;

enum _VEHICLE_MODE {
	NONE,
	AUTO_PARK,
	AUTO_RETRIEVE,
	REMOTE_DRIVE
};

enum _SONAR_LIST {
	LEFT_SONAR = 1,
	FRONT_SONAR,
	RIGHT_SONAR,
	REAR_SONAR
};

enum _SERVO {
	STEERING_SERVO,
	DOOR_SERVO
};

/*-----------------------------------------------
*/

uint8_t no[] = "9791010426";		//
emb_sys
uint8_t msg[] = "Vehicle Motion Alert";
uint8_t bti_msg[] = "bti say remote motion
alert, remote motion alert, remote motion
alert";

uint32_t vehicle_forward_motion_cnt = 0;

/*-----------------------------------------------
*/

#define ACCEL_THRESH	10

#define SIDE_OBJ_DIST_MAX	50
	// cm
#define FRONT_OBJ_DIST_MAX	50
	// cm
#define REAR_OBJ_DIST_MAX	20
	// cm

#define SIDE_GAP_CNT_TIME
	1600	// milli sec
#define	SIDE_GAP_SAMPLE_TIME
	200		// milli sec

#define SIDE_GAP_CNT_MAX
	SIDE_GAP_CNT_TIME		/
SIDE_GAP_SAMPLE_TIME
#define	SIDE_GAP_TIMEOUT_MAX
	50

// parking maneuver period in milli sec
#define LEFT_TURN_TIME
		2200	//1800
#define RIGHT_TURN_TIME
			2200	//1800
#define
REVERSE_TIME_ALONG_PATH
	800
#define	REVERSE_TIME_INTO_GAP
		3000
#define
FORWARD_TIME_OUT_OF_GAP 3200

/* Servo motor positions */
#define POSITION_RIGHT		4.00f
#define POSITION_MIDDLE
	7.50f
#define POSITION_LEFT
	11.00f
#define POSITION_OPEN		4.00f
#define POSITION_CLOSE
	11.00f
#define POSITION_LOW_POWER	0

/*-----------------------------------------------
*/

static void buzzer_on(void) {
	GPIO0_SetBit(1 << 26);
}
/*-----------------------------------------------
*/

static void buzzer_off(void) {
	GPIO0_ClrBit(1 << 26);
}
/*-----------------------------------------------
*/

static void buzzer_init(void) {
	/**
	 * P0.26 (AD5) is buzzer pin
	 * Remember: DAC is not available in LPC1764
	 */
	GPIO0_SetDir((1 << 26), GPIO_OUT);
	buzzer_on();
	buzzer_off();
}
/*-----------------------------------------------
*/

static void servo_init(void) {

	/* Initialize servo PWM */
	pwm_init((PWM0_ENABLE | PWM1_ENABLE), 0, 50);

	/* Reset to mid position */
	pwm_duty_cycle_frac(STEERING_SERVO, POSITION_MIDDLE);
	pwm_duty_cycle_frac(DOOR_SERVO, POSITION_MIDDLE);
	delayMs(0, 1000);

	/* Stop power, if any */
	pwm_duty_cycle_frac(STEERING_SERVO, POSITION_LOW_POWER);
	pwm_duty_cycle_frac(DOOR_SERVO, POSITION_LOW_POWER);
}
/*-----------------------------------------------
*/

static void servo_control(uint32_t servo, float position, uint32_t delay_ms) {

	pwm_duty_cycle_frac(servo, position);
	vTaskDelay(delay_ms);
	pwm_duty_cycle_frac(servo, POSITION_LOW_POWER);
}
/*-----------------------------------------------
*/

static void prvSetupHardware(void) {

	/* Initialize delay_timer, board_leds and board_keys */
	MBv14_BoardInit(1);

	/* Initialize Servo Motors */
	servo_init();

	/* Initialize Buzzer */
	buzzer_init();

	/* Initialize Robot Motors */
	wheel_init();

	/* Initialize GLCD */
	glcd_init();

	/*
	 * Initialize GSM modem on UART0
	 * baud rate = 9600
	 */
	gsm_init();

	/*
	 * Initialize UART3
	 * used as Bluetooth modem serial port
	 * baud rate = 9600
	 */
	xUART3PortInit(9600, 64);


	/* Initialize LIS302DL */
	lis302dl_init(LOW_G);

	/* Initialize SONAR */
	sonar_init(SONAR1_ENABLE);
	sonar_init(SONAR2_ENABLE);
	sonar_init(SONAR3_ENABLE);
	sonar_init(SONAR4_ENABLE);

	/* Initialize Timer1 for periodic interrupt */
	init_interrupt_timer(1, TIME_INTERVAL);

}
/*-----------------------------------------------
*/

static void
```

```
get_sonar_snapshot(SONAR_TYPE
*sonar) {

        sonar_get_distance(FRONT_SON
AR, &sonar->front);
        sonar_get_distance(RIGHT_SONA
R, &sonar->right);
        sonar_get_distance(LEFT_SONAR
, &sonar->left);
        sonar_get_distance(REAR_SONA
R, &sonar->rear);

        if ((sonar->front > 100) || (sonar-
>front == 0))
                sonar->front = 100;
        if ((sonar->right > 100) || (sonar-
>right == 0))
                sonar->right = 100;
        if ((sonar->left > 100) || (sonar-
>left == 0))
                sonar->left = 100;
        if ((sonar->rear > 100) || (sonar-
>rear == 0))
                sonar->rear = 100;
}
/*------------------------------------------*/

static                              void
put_sonar_snapshot(SONAR_TYPE
*sonar) {

        xGLCDMsg xMsg;

        vGLCDMsgWrite(2,              4,
UNSIGNED_NUMBER_TYPE,       (void
*)(uint32_t) sonar->front, &xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
        vGLCDMsgWrite(3,              8,
UNSIGNED_NUMBER_TYPE,       (void
*)(uint32_t) sonar->right, &xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
        vGLCDMsgWrite(3,              0,
UNSIGNED_NUMBER_TYPE,       (void
*)(uint32_t) sonar->left, &xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
        vGLCDMsgWrite(4,              4,
UNSIGNED_NUMBER_TYPE,       (void
*)(uint32_t) sonar->rear, &xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
}
/*------------------------------------------*/

static                              void
wheel_reverse_and_rear_scan(uint32_t
delay_ms) {
        SONAR_TYPE sonar;

        uint32_t cnt = delay_ms / 200;
        uint32_t i;

        wheel_reverse();

        for (i = 0; i < cnt; ) {

                vTaskDelay(200);

        get_sonar_snapshot(&sonar);

        put_sonar_snapshot(&sonar);

                if        (sonar.rear        <
REAR_OBJ_DIST_MAX) {
                        wheel_stop();

                } else {
                        wheel_reverse();
                        ++i;
                }
        }
}
/*------------------------------------------*/

static                              void
parking_maneuver(BTI_COMMAND
parking_dir) {
        buzzer_on();
        wheel_stop_delay(2000);
        buzzer_off();

//
        wheel_reverse_delay(REVERSE_
TIME_ALONG_PATH);
```

```
        wheel_reverse_and_rear_scan(RE
VERSE_TIME_ALONG_PATH);
        wheel_stop_delay(1000);

        if (parking_dir == PARK_RIGHT)
{

        servo_control(STEERING_SERV
O, POSITION_RIGHT, 1000);

        wheel_left_turn_delay(LEFT_TUR
N_TIME);
        } else {

        servo_control(STEERING_SERV
O, POSITION_LEFT, 1000);

        wheel_right_turn_delay(RIGHT_T
URN_TIME);
        }
        wheel_stop_delay(1000);

        servo_control(STEERING_SERV
O, POSITION_MIDDLE, 1000);
//
        wheel_reverse_delay(REVERSE_
TIME_INTO_GAP);
        wheel_reverse_and_rear_scan(RE
VERSE_TIME_INTO_GAP);

        buzzer_on();
        wheel_stop_delay(1000);
        buzzer_off();
}
/*-----------------------------------------*/

static void auto_park(BTI_COMMAND
parking_dir) {

        xGLCDMsg xMsg;

        uint8_t parking_gap;
        uint8_t gap_cnt;
        uint8_t timeout;
        uint8_t front_obj_detected;

        SONAR_TYPE sonar;

        vUART3PutString("say auto
parking started.");

        servo_control(STEERING_SERV
O, POSITION_MIDDLE, 1000);
        wheel_forward();    //    start
moving...

        gap_cnt = 0;
        parking_gap = 0;
        front_obj_detected = 0;
        timeout                    =
SIDE_GAP_TIMEOUT_MAX;

        do {
                vTaskDelay(200);

        get_sonar_snapshot(&sonar);

        put_sonar_snapshot(&sonar);

                if    (sonar.front    <
FRONT_OBJ_DIST_MAX) {
                        if
(!front_obj_detected) {

        front_obj_detected = 1;

        wheel_stop();
                        }
                        continue;

                } else {
                        if
(front_obj_detected) {

        wheel_forward();

        front_obj_detected = 0;
                        }
                }


        ++vehicle_forward_motion_cnt;
                --timeout;
                if (timeout == 0) {
                        break;
                }

                uint16_t side_obj_dist;
                if    (parking_dir    ==
PARK_RIGHT)    side_obj_dist    =
sonar.right;
                else
        side_obj_dist = sonar.left;

                if (side_obj_dist >=
SIDE_OBJ_DIST_MAX) {
                        gap_cnt++;
                        if (gap_cnt >=
SIDE_GAP_CNT_MAX) {
                                parking_gap
= 1;
                        }
                } else {
                        if (gap_cnt) {
                                buzzer_on();

        vTaskDelay(50);
                                buzzer_off();

                                gap_cnt = 0;
                        }
                }

                vGLCDMsgWrite(5, 0,
UNSIGNED_NUMBER_TYPE, (void
*)(uint32_t) gap_cnt, &xMsg);

        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
                vTaskDelay(1);

        } while (!parking_gap);

        if (parking_gap) {      // parking
gap found

                vUART3PutString("say
parking space found");

                vGLCDMsgWrite(5, 6,
STRING_TYPE, (void *) "gap true",
&xMsg);

        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
                vTaskDelay(1);


        parking_maneuver(parking_dir);
```

```
                vUART3PutString("say
auto parking complete");


        } else {        // no parking gap
found
                vGLCDMsgWrite(5, 6,
STRING_TYPE, (void *) "gap fail",
&xMsg);

        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
                vTaskDelay(1);

                vUART3PutString("say
sorry. parking space could not be found.");

                buzzer_on();
                vTaskDelay(4000);
                buzzer_off();
        }

        return;
}
/*-----------------------------------------------
*/

void auto_retrieve(BTI_COMMAND
parking_dir) {

        SONAR_TYPE sonar;

        vUART3PutString("say auto
retrieving started");

        servo_control(STEERING_SERV
O, POSITION_MIDDLE, 1000);
        wheel_forward_delay(FORWARD
_TIME_OUT_OF_GAP);
        wheel_stop_delay(1000);

        if (parking_dir == PARK_RIGHT)
{

        servo_control(STEERING_SERV
O, POSITION_LEFT, 1000);

        wheel_left_turn_delay(LEFT_TUR
N_TIME);
```

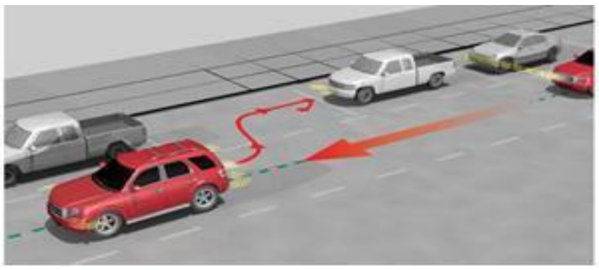
```
        } else {

        servo_control(STEERING_SERV
O, POSITION_RIGHT, 1000);

        wheel_right_turn_delay(RIGHT_T
URN_TIME);
        }
        wheel_stop_delay(1000);

        servo_control(STEERING_SERV
O, POSITION_MIDDLE, 1000);
        wheel_forward();

        while
(vehicle_forward_motion_cnt) {

                vTaskDelay(200);

        get_sonar_snapshot(&sonar);

        put_sonar_snapshot(&sonar);

                if      (sonar.front      <
FRONT_OBJ_DIST_MAX) {


        vUART3PutString("say    obstacle
detected");


        wheel_stop_delay(1000);


        servo_control(STEERING_SERV
O, POSITION_RIGHT, 1000);

        wheel_right_turn_delay(RIGHT_T
URN_TIME);

        wheel_stop_delay(1000);


        servo_control(STEERING_SERV
O, POSITION_MIDDLE, 1000);

        wheel_forward_delay(2000);

        wheel_stop_delay(1000);

        servo_control(STEERING_SERV
O, POSITION_LEFT, 1000);

        wheel_left_turn_delay(LEFT_TUR
N_TIME);

        wheel_stop_delay(1000);


        servo_control(STEERING_SERV
O, POSITION_MIDDLE, 1000);

        wheel_stop_delay(1000);

                        wheel_forward();
                }

                if
(vehicle_forward_motion_cnt)
                        --
vehicle_forward_motion_cnt;
        }

        wheel_stop();
        vUART3PutString("say        auto
retrieving complete");

        return;
}
/*-----------------------------------------------
*/

void remote_drive(void) {

        BTI_COMMAND bti_cmd;
        xGLCDMsg xMsg;

        vUART3PutString("say remote
drive started");

        /* update the glcd */
        vGLCDMsgWrite(1, 6,
STRING_TYPE, (void *) "      ",
&xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
        vGLCDMsgWrite(1, 6,
STRING_TYPE, (void *) "iDrive",
&xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);

        servo_control(STEERING_SERV
O, POSITION_MIDDLE, 1000);
        uint32_t servo_position =
POSITION_MIDDLE;

        while (1) {

                /* Receive the BTI
Command */

        xQueueReceive(xBtiCmdQueue,
&bti_cmd, portMAX_DELAY);
                if (bti_cmd ==
FORWARD) {wheel_forward();
                        if (servo_position !=
POSITION_MIDDLE)
{servo_control(STEERING_SERVO,
POSITION_MIDDLE, 500);
servo_position = POSITION_MIDDLE;
        }

        } else if (bti_cmd == REVERSE)
{wheel_reverse();
        if (servo_position
!=POSITION_MIDDLE)
{servo_control(STEERING_SERVO,
POSITION_MIDDLE, 500);
        servo_position =
POSITION_MIDDLE;
        }
        } else if (bti_cmd == RIGHT)
{wheel_right_turn();
        if (servo_position !=
POSITION_RIGHT)
{servo_control(STEERING_SERVO,
POSITION_RIGHT, 500);
servo_position = POSITION_RIGHT;
                        }

        } else if (bti_cmd == LEFT)
{wheel_left_turn();
        if (servo_position !=
POSITION_LEFT)
{servo_control(STEERING_SERVO,
POSITION_LEFT, 500);
        servo_position =
POSITION_LEFT;
        }
        } else if (bti_cmd == STOP) {
        wheel_stop();
        if (servo_position !=
POSITION_MIDDLE)
{servo_control(STEERING_SERVO,
POSITION_MIDDLE, 500);
        servo_position =
POSITION_MIDDLE;
        }
        } else if (bti_cmd ==
SMART_DRIVE) {vGLCDMsgWrite(1,
6, STRING_TYPE, (void *) "        ",
&xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
vUART3PutString("say remote drive
ended");
//exit this function
break;
}
}
}
/*-----------------------------------------------
*/
void display_bti_cmd(uint8_t
*bti_cmd_ptr) {xGLCDMsg xMsg;
        /* update the glcd */
        vGLCDMsgWrite(0, 5,
STRING_TYPE, (void *) "        ",
&xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
        vGLCDMsgWrite(0, 5,
STRING_TYPE, (void *) bti_cmd_ptr,
&xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
}
/*-----------------------------------------------
*/
/*
 * Receive vehicle movement commands
over Bluetooth
 * and sends it to other tasks using a queue
 */
void vBluetoothTask(void *pvParameters)
{
        xGLCDMsg xMsg;
        BTI_COMMAND bti_cmd;
        portCHAR bti_string[20];
        uint8_t i;
        uint8_t bti_string_received;
        portCHAR rx_char;
        xBtiCmdQueue = xQueueCreate(4,
sizeof(BTI_COMMAND));
        /* Display "CMD:" title string in
GLCD */
        vGLCDMsgWrite(0,              0,
STRING_TYPE,   (void   *)   "CMD:",
&xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(1);
        while (1) {
                i = 0;
                bti_string_received = 0;
                memset(bti_string, '\0', 20);
                // wait for command string
send from btinterface application
                do {

        if(xUART3GetChar(&rx_char,
portMAX_DELAY) == pdTRUE) {
                if (rx_char == '\r') {
                bti_string[i] = '\0';
                bti_string_received = 1;
                } else {
                bti_string[i] = rx_char;
                i++;
                }
                }
                }                   while
(!bti_string_received);
                if (strcmp((const char *)
bti_string, "btinterface") == 0) {bti_cmd =
BTI_CONNECTED;
// customize btinterface appearance
vUART3PutString("reset;ht        Android
Car;s1t  Android  Car;b1  Park_Right;b2
Park_Left;b3                 Retrieve;b4
Remote_Drive;sbh;screen1");
strcpy((char *)bti_string, "Connected");
} else if (strcmp((const char *) bti_string,
"up") == 0) {
                bti_cmd = FORWARD;
                } else if (strcmp((const char
*) bti_string, "down") == 0) {bti_cmd =
REVERSE;
                } else if (strcmp((const char
*) bti_string, "right") == 0) {bti_cmd =
RIGHT;
                } else if (strcmp((const char
*) bti_string, "left") == 0) {
                        bti_cmd = LEFT;
                } else if (strcmp((const char
*) bti_string, "fire") == 0) {
                        bti_cmd = STOP;
                } else if (strcmp((const char
*) bti_string, "b1") == 0) {
                        bti_cmd          =
PARK_RIGHT;
                } else if (strcmp((const char
*) bti_string, "b2") == 0) {
                        bti_cmd          =
PARK_LEFT;
                } else if (strcmp((const char
*) bti_string, "b3") == 0) {
                        bti_cmd          =
RETRIEVE;
                } else if (strcmp((const char
*) bti_string, "b4") == 0) {
                        bti_cmd          =
SMART_DRIVE;
                } else if (strcmp((const char
*) bti_string, "open") == 0) {
                        bti_cmd          =
DOOR_OPEN;
                } else if (strcmp((const char
*) bti_string, "close") == 0) {
                        bti_cmd          =
DOOR_CLOSE;
                }

                display_bti_cmd((uint8_t *)
bti_string);

                // send the command using
Queue

        xQueueSendToBack(xBtiCmdQue
ue, &bti_cmd, portMAX_DELAY);

        }
}
```

```
/*-----------------------------------------------
*/
/*
 * read accelerometer values and send sms
if motion is abnormal
 */
void vRemoteAlertTask(void
*pvParameters) {

        ACCELEROMETER_T accel,
accel_ref;

        uint8_t run_state = 0;
        uint8_t alert_flag = 0;

        /* Take the semaphores once */
        xSemaphoreTake(xRemoteAlertTa
skBeginSemaphore, 0);
        xSemaphoreTake(xRemoteAlertTa
skEndSemaphore, 0);

        while (1) {

                /* wait for the semaphore
signal */

        xSemaphoreTake(xRemoteAlertTa
skBeginSemaphore, portMAX_DELAY);
                run_state = 1;

                // read the ref. position

        lis302dl_block_read(&accel_ref);

                while (run_state) {

                        vTaskDelay(200);

                        if (!alert_flag) {

        lis302dl_block_read(&accel);

                                if ((accel.x >
(accel_ref.x + ACCEL_THRESH)) ||
(accel.x < (accel_ref.x -
ACCEL_THRESH)) ||

        (accel.y > (accel_ref.y +
ACCEL_THRESH)) || (accel.y <
(accel_ref.y - ACCEL_THRESH)) ||

        (accel.z > (accel_ref.z +
ACCEL_THRESH)) || (accel.z <
(accel_ref.z - ACCEL_THRESH))) {


        GPIOSetBit(0, 22);

        buzzer_on();


        gsm_send_sms(no, msg);


        GPIOClrBit(0, 22);

        buzzer_off();


        gsm_send_sms(no, bti_msg);


        alert_flag = 1;

                                }
                        }

                        if
(xSemaphoreTake(xRemoteAlertTaskEnd
Semaphore, 0) == pdPASS) {
                                run_state =
0;
                        }

                }// run_state

        }// while

}
/*-----------------------------------------------
*/
/*
 * managing task
 */
void vMainTask(void *pvParameters) {

        BTI_COMMAND bti_cmd;
        xGLCDMsg xMsg;
        BTI_COMMAND parking_dir =
PARK_RIGHT;         // default right
```

```
        /* Display "MODE:" title string in
GLCD */
        vGLCDMsgWrite(1,              0,
STRING_TYPE,  (void  *)  "MODE:",
&xMsg);
        xQueueSendToBack(xGLCDQueu
e, &xMsg, 0);
        vTaskDelay(100);

        while (1) {

                /*   Receive   the   BTI
Command */

        xQueueReceive(xBtiCmdQueue,
&bti_cmd, portMAX_DELAY);

                if       (bti_cmd       ==
PARK_RIGHT) {

                        parking_dir       =
bti_cmd;

        auto_park(PARK_RIGHT);


        xSemaphoreGive(xRemoteAlertTa
skBeginSemaphore);  // begin remote alert

                }   else   if   (bti_cmd   ==
PARK_LEFT) {

                        parking_dir       =
bti_cmd;

        auto_park(PARK_LEFT);


        xSemaphoreGive(xRemoteAlertTa
skBeginSemaphore);  // begin remote alert

                }   else   if   (bti_cmd   ==
RETRIEVE) {


        xSemaphoreGive(xRemoteAlertTa
skEndSemaphore);    // end remote alert

        auto_retrieve(parking_dir);

                }   else   if   (bti_cmd   ==
SMART_DRIVE) {

                        remote_drive();

                }   else   if   (bti_cmd   ==
DOOR_OPEN) {


        servo_control(DOOR_SERVO,
POSITION_OPEN, 1000);

                }   else   if   (bti_cmd   ==
DOOR_CLOSE) {


        servo_control(DOOR_SERVO,
POSITION_CLOSE, 1000);

                }
        }
}
/*-------------------------------------------------
*/

int main(void) {

        /* Let us start (with) hardware first
*/
        prvSetupHardware();

        /* Create the semaphores */
        vSemaphoreCreateBinary(xRemote
AlertTaskBeginSemaphore);
        vSemaphoreCreateBinary(xRemote
AlertTaskEndSemaphore);

        /* Start the tasks */
        vStartGLCDGateKeeperTask();
        xTaskCreate(vBluetoothTask,     (
signed    portCHAR    *    )    "BT",
configMINIMAL_STACK_SIZE    *    2,
NULL, tskIDLE_PRIORITY + 1, NULL);
        xTaskCreate(vMainTask,  (  signed
portCHAR        *        )        "Main",
configMINIMAL_STACK_SIZE    *    2,
NULL, tskIDLE_PRIORITY + 1, NULL);
        xTaskCreate(vRemoteAlertTask,  (
signed    portCHAR    *    )    "Alert",
configMINIMAL_STACK_SIZE * 2,
NULL, tskIDLE_PRIORITY + 1, NULL);

        /* Start the scheduler */
        vTaskStartScheduler();

        /* Will only get here if there was
insufficient memory to create the idle task
*/
        for (;;);

        return 0;
}
/*-------------------------------------------------
*/

#if
configCHECK_FOR_STACK_OVERFLO
W > 0
void
vApplicationStackOverflowHook(xTaskH
andle *pxTask, signed char
*pcTaskName) {

        /* This function will get called if a
task overflows its stack. */
        (void) pxTask;
        (void) pcTaskName;

        for (;;);
}
#endif
/*-------------------------------------------------
*/
```

III. AUTONOMOUS SELF RETRIEVING

We all know that retrieving a car from the parking lot is more difficult task when compared to parking it. So many problems will be faced at the congested parking area while retrieving the car. One of the problems may be that the other users may not have parked the car correctly in the particular slot .which makes the user feeling difficult in retrieving the car out of the parking space. Hence in order to overcome this issue the retrieve mode is selected through BT interface app which makes the car unit to make it available near the users place by Bluetooth connectivity. One push of the button is all it takes. The car drives it into the user's place, as shown in fig.2.which makes it very convenient for the user in retrieving the car.



(a)

(b)

Fig.2 overall appearance of autonomous self-retrieving system (a) Side View (b) Front View.

IV. REMOTE DOOR OPEN/CLOSE CONTROL

In order to perform the remote door open/close operation the user has to go to the terminal window of the BT interface application and has to send the open command for remote door opening and close command for door close operation. Once the command is sent from the app the controller receives the data and makes the car unit perform opening or closing of the door based on the respective command received.

V. CONCLUSION

Intelligent autonomous android car unit have been implemented by using the embedded controller and free RTOS technology through BT interface android application and Bluetooth communication. The controller is capable of parking the car in an appropriate parking space effectively by integrating the sensor data in park mode and has the ability to make the vehicle to avoid collisions to ensure safe parking. The car is successfully retrieved in retrieve mode and door/close operation is performed when open/close command is sent through the terminal window of the android application.

VI. ACKNOWLEDGEMENT

The authors would like to thank the Management and Department of Mechatronics, Jeppiaar engineering College for their continuous support during the tenure of the work.

VII. REFERENCES

[1] Jae Kyu Suhr and Ho Gi Jung, “Sensor Fusion-Based Vacant Parking Slot Detection and Tracking” IEEE Transactions on Intelligent Transportation Systems,Vol.15,No.1,Febrauary 2014.

[2] David Herrero, Jorge Villagrá, and Humberto Martínez, “Self-Configuration of Waypoints for Docking Maneuvers of Flexible Automated Guided Vehicles”IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, VOL. 10, NO. 2, APRIL 2013.

[3] Joey Wilson and Neal Patwari, “A fade level skew laplace signal strength model for device free localization with wireless network”, IEEE Trans Mobile Computing, Vol. 11, No. 6, June 2012.

[4] Ho Gi Jung, Chi Gun Choi, Pal Joo Yoon, Jaihie Kim “Semi-automatic Parking System Recognizing Parking Lot Markings”.

[5] Zhang Bin, Jiang Dalin, Wang Fang, Wan Tingting “A design of parking space detector based on video image”International Conference on Electronic Measurement and Instruments - ICEMI , 2009.

[6] Sungon Lee, Youngil Youm, Wan Kyun Chung, Wankyun Chung “Control of a Car-Like Mobile Robot for Parking Problem”International Conference on Robotics and Automation - ICRA, vol. 1, pp. 1-6, 1999.

[7] S.C.Hanche, Pooja Munot,Pranali Bagal, Kirti Sonawane, Pooja Pise, “Automated Vehicle Parking System using RFID”, ITSITEEE, ISSN(PRINT) :2320-8945,Volume 1,Issue-2,2013 .

[8] Sheng-Fuu Lin, Yung-Yao Chen, “A visionbased parking lot management system”, 2006 IEEE Conference on Systems, Man, and Cybernetics. Peter. C, pp. 2897- 902, Oct. 2006.

[9] K. An, J. Choi, and D. Kwak, “Automatic valet parking system Incorporating a Nomadic device and parking servers,” in *Proc. IEEE Int. Conf. Consum. Electron.* Jan. 2011, pp. 111–112.

[10] K. Sung, J. Choi, and D. Kwak, “Vehicle control system for automatic valet parking with infrastructure sensors,” in *Proc. IEEE Int. Conf. Consum.Electron.* Jan. 2011, pp. 567– 56