به نام پروردگار زیبایی

گزارش پروژه درس نهفته

امین گرامی مهر

محمدعرفان زارع زردینی

۱۴۰۲زمستان

دانشگاه علم و صنعت ایران

نام پروژه: ردیاب خودرو با mc60

توضیحات اولیه:

پروژه پیرامون طراحی پروگرم نمودن سخت افزار و فرستادن لوکیشن مکانی آن از طریق سیم کارت می باشد.ما برای این کار از اینترنت نسل دوم(gsm) استفاده نمودیم و سخت افزار مورد استفاده ما mc60 بود.لوکیشن از طریق آنتن ها دریافت، تبدیلات لازم برروی آن انجام شده و در نهایت هم به وسیله gsm ، به مقصد مشخص شده ارسال می شود.

چالش ها:

1) عدم پیدا شدن سخت افزار در برخی از بازارهای تهران و ناموجود بودن در سایت های معتبر
2) آشنا نبودن با پروتکل mqtt برای پروژه
3) عدم اطلاع از ساختار توابع و دستورات مربوط به سخت افزار که سبب می شد در هنگام نوشتن برنامه به باگ بخوریم.
4) خراب بودن سخت افزار اولیه
5) پروگرم نشدن سخت افزار اولیه به دلایلی نامعلوم
6) عدم امکان دیباگ کردن کد به صورت عملی و داخل کد که یافتن ایراد کد را سخت می نمود.
7) یکسان نبودن فرمت داده های تشخیص داده شده توسط آنتن و ورودی گوگل مپ
8) کاربرد نداشتن مستقیم برخی دستورات c در کد و ایجاد ارور(باید از دستورات مشابه در کتابخانه opencpu استفاده شود یا تابع کارکرد آن دستی نوشته شود.)

توضیحات پروژه:

در این پروژه برنامه در جهت پوشش دادن خواسته ها و نیاز های مورد نظر براساس دیتا های دریافتی مورد نظر، نوشته شده و درنهایت با برنامه q flash برروی سخت افزار انتقال داده می شود. همچنین در جهت بررسی داده های خروجی نهایی و صحت آن با استفاده از برنامه mqttx ، و ست کردن topic آن، و با بروکر (ما اینجا از mosquitto استفاده نمودیم) انجام دادیم.

برای پیاده سازی، توابعی که ما استفاده می کنیم باید به صورت موازی اجرا شوند و داده ها را پردازش و خروجی دهند . پس در فایل custom_task_cfg.h ، تسک ها را تعریف نموده و اطلاعات آن را ست می کنیم که به شرح و ساختار زیر می باشد:

```
/*-------------------------------------------------------------------------------------------------------
|        Task Entry Function | Task Id Name    | Task Stack Size (Bytes) | Default Value1 | Default Value2 |
 *-------------------------------------------------------------------------------------------------------*/
TASK_ITEM(proc_main_task,       main_task_id,    10*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
TASK_ITEM(proc_reserved1,       reserved1_id,    5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
TASK_ITEM(proc_reserved2,       reserved2_id,    5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
TASK_ITEM(proc_subtask1,        subtask1_id,     5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
TASK_ITEM(proc_subtask2,        subtask2_id,     10*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
TASK_ITEM(proc_subtask3,        subtask3_id,     5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
```

برای بخش gps و gsm ساختار زیر پیاده شده است:

ابتدا باید چک شود مقعیت مکانی پیدا شده است یا خیر. پس پیدا شدن یا نشدن آن را با یک led روی سخت افزار نمایش می دهیم . حال براساس پیام دریافتی، ابتدا RIL وارد مرحله ready می شود و تعریف اولیه می شود، و gps فعال میشه. همچنین در تابع اولیه شرایط و سخت افزار ها رو هم چک میکند که درست کار کند و درست باشد.(شامل سیم کارت،mqtt client، شبکه، topic و...) همچنین مراحل دریافت اطلاعات و تبادلات در بروکر. درنهایت با اوکی بودن این موارد ، شبکه بررسی شده و در صورت مهیا بودن شرایط، رجیستر می شود. همچنین برای سیمکارت باید در برنامه باید APN درست به همراه یوزرنیم و پسورد مناسب، ست شود.

تکه ای از کد زده شده در زیر قرار داده شده است.

```
case URC_GSM_NW_STATE_IND:
    APP_DEBUG("GSM Network Status:%d\r\n", msg.param2);
    break;
case URC_SIM_CARD_STATE_IND:
{
    APP_DEBUG("//<SIM Card Status:%d\r\n", msg.param2);
    if (SIM_STAT_READY == msg.param2)
    {
        Ql_Timer_Start(MQTT_TIMER_ID, MQTT_TIMER_PERIOD, TRUE);
        APP_DEBUG("//<state timer start,ret = %d\r\n", ret);
    }
}
break;
case URC_MQTT_OPEN:
{
    mqtt_urc_param_ptr = msg.param2;
    if (0 == mqtt_urc_param_ptr->result)
    {
        APP_DEBUG("//<Open a MQTT client successfully\r\n");
        m_mqtt_state = STATE_MQTT_CONN;
    }
    else
    {
        APP_DEBUG("//<Open a MQTT client failure,error = %d\r\n", mqtt_urc_param_ptr->result);
    }
}
break;
case URC_MQTT_CONN:
{
    mqtt_urc_param_ptr = msg.param2;
    if (0 == mqtt_urc_param_ptr->result)
    {
        APP_DEBUG("//<Connect to MQTT server successfully\r\n");
        m_mqtt_state = STATE_MQTT_SUB;
    }
    else
    {
        APP_DEBUG("//<Connect to MQTT server failure,error = %d\r\n", mqtt_urc_param_ptr->result);
```

حال آمده ایم تسک موجود را ریز نموده به زیر تسک برده و برای آنها تابع نوشته ایم.تابع زیر برای خواندن سریال است و شروع برنامه است.

```
void proc_subtask1(s32 TaskId)
{
    s32 ret;
    ST_MSG msg;
    ST_UARTDCB dcb;
    Enum_SerialPort mySerialPort = UART_PORT1;
    dcb.baudrate = 115200;
    dcb.dataBits = DB_8BIT;
    dcb.stopBits = SB_ONE;
    dcb.parity = PB_NONE;
    dcb.flowCtrl = FC_NONE;
    Ql_UART_Register(mySerialPort, CallBack_UART_Hdlr, NULL);
    Ql_UART_OpenEx(mySerialPort, &dcb);
    Ql_UART_ClrRxBuffer(mySerialPort);
    APP_DEBUG("START PROGRAM \r\n");
    while (TRUE)
    {
        Ql_OS_GetMessage(&msg);
        switch (msg.message)
        {
        case MSG_ID_USER_START:
            break;
        default:
            break;
        }
    }
}
```

برنامه زیر برای آن است که داده مکانی آپدیت شود و اطلاعات جدید دریافت شود و هر 1 ثانیه بروز می شود.

همنین RIL را چک میکنیم که ready شده یا خیر. همچنین اگر موقعیت به دست آمده بود ، برای نمایش سخت افزاری، led مان چند بار چشمک میزند(2 بار اینجا) درحالیکه اگر اوکی نبود، یک بار چشمک می زند.

```
proc_subtask2(s32 TaskId)
{
    while (1)
    {
        Ql_Sleep(1000);
        if (RIL_STATUS = 1)
        {
            iRet = RIL_GPS_Read("RMC", RMC_BUFFER);
            if (RIL_AT_SUCCESS != iRet)
            {
                APP_DEBUG("Read %s information failed.\r\n", "RMC");
            }
            else
            {
                if (RMC_BUFFER[30] == 'A')
                {
                    Ql_GPIO_SetLevel(LED_1, PINLEVEL_HIGH);
                    Ql_Sleep(50);
                    Ql_GPIO_SetLevel(LED_1, PINLEVEL_LOW);
                    Ql_Sleep(50);
                    Ql_GPIO_SetLevel(LED_1, PINLEVEL_HIGH);
                    Ql_Sleep(50);
                    Ql_GPIO_SetLevel(LED_1, PINLEVEL_LOW);
                }
                else if (RMC_BUFFER[30] == 'V')
                {
                    Ql_GPIO_SetLevel(LED_1, PINLEVEL_HIGH);
                    Ql_Sleep(50);
                    Ql_GPIO_SetLevel(LED_1, PINLEVEL_LOW);
                }
            }
        }
    }
}
```

در تابع GPSPower ، اگر ورودی داده ششده 1 باشد، gps روشن است و درصورتی که قبل آن خاموش بوده باشد، روشن می شود . در صورتی هم که ورودی 0 باشد، در صورت روشن بودن ، خاموش می کند و درغیر این صورت، اطلاع میدهد که خاموش است.

```
void GPSPower(int status)
{
    if (status == 1)
    {
        iRet = RIL_GPS_Open(1);
        if (RIL_AT_SUCCESS != iRet)
        {
            APP_DEBUG("GPS is on \r\n");
        }
        else
        {
            APP_DEBUG("Power on GPS Successful.\r\n");
        }
    }
    else if (status == 0)
    {
        iRet = RIL_GPS_Open(0);
        if (RIL_AT_SUCCESS != iRet)
        {
            APP_DEBUG("GPS is off \r\n");
        }
        else
        {
            APP_DEBUG("Power off GPS Successful.\r\n");
        }
    }
}
```

تابع زیر نیز پورت سریال را دیتایش را خوانده و دیتا خواسته و مورد بررسی را برمی گرداند.

همچنین در تابع CallBack_UART_Hdlr ، هم ما بر اساس دیتا دریافتی بافر، و بررسی آن gpsرا روشن نموده یا خاموش می کنیم یا لوکیشن را بررسی می کنیم.

```
static s32 ReadSerialPort(Enum_SerialPort port, /*[out]*/ u8 *pBuffer, /*[in]*/ u32 bufLen)
{
    s32 rdLen = 0;
    s32 rdTotalLen = 0;
    if (NULL == pBuffer || 0 == bufLen)
    {
        return -1;
    }
    Ql_memset(pBuffer, 0x0, bufLen);
    while (1)
    {
        rdLen = Ql_UART_Read(port, pBuffer + rdTotalLen, bufLen - rdTotalLen);
        if (rdLen <= 0)
        {
            break;
        }
        rdTotalLen += rdLen;
    }
    return rdTotalLen;
}
```

```
static void CallBack_UART_Hdlr(Enum_SerialPort port, Enum_UARTEventType msg, bool level, void *customizedPara)
{
    switch (msg)
    {
    case EVENT_UART_READY_TO_READ:
    {
        char *p = NULL;
        s32 totalBytes = ReadSerialPort(port, m_RxBuf_Uart, sizeof(m_RxBuf_Uart));
        if (totalBytes <= 0)
        {
            break;
        }
        if (Ql_strstr(m_RxBuf_Uart, "GPSOn"))
        {
            APP_DEBUG("ok\r\n");
            GPSPower(1);
            break;
        }
        if (Ql_strstr(m_RxBuf_Uart, "GPSOff"))
        {
            APP_DEBUG("ok\r\n");
            GPSPower(0);
            break;
        }
        if (Ql_strstr(m_RxBuf_Uart, "location"))
        {
            APP_DEBUG("ok\r\n");
            APP_DEBUG("%s \r\n", RMC_BUFFER);
            break;
        }
        break;
```

به صورت کلی براساس پالس داده ای که gps با آنتن، دریافت نموده و به سخت افزار داده است، پردازش نموده و تبدیلات لازم را انجام داده ایم و به بروکر داده ایم.

بعد از دریافت دیتا اولیه، باتوجه به اینکه دیتا دریافتی خام است و باید تغییرات لازم انجام شود تا به فرمت درست برسد، برروی آن تغییرات انجام میپذیرد. همچنین چون دیتای دریافتی ابتدا فعال شدن gpsدرست نیست، نیاز هست که کالیبره شود و درحال تغییر است و بعد از چند دقیقه به حدود دقت مناسبی میرسد .

دیتا دریافتی به حالت csv و شامل 13 بخش است که هرکدام مفهوم و ساختاری را نشان میدهند که می شود به طول و عرض جغرافیایی، جهت مختصاتی(شمال، جنوب ، شرق، غرب) ، سرعت، درجه و مانند آن می باشد.ما میآییم براساس اینکه میان هر کدام از این 13 بخش ویرگول قرار گرفته، عناصر آن را جدا می نماییم. سپس طول و عرض جغرافیایی را جدا نموده و تبدیلات مورد نیاز که شامل جابجایی نماد کسر(/) با تقسیم بر 100 نمودن فرمت عددی آن است انجام میدهیم. با توجه به اینکه دیتا داده شده با فرمت Degrees Minutes Seconds می باشد و دیتایی که باید در فرمت url باشد ، Decimal Degrees نام دارد باید تبدیلات انجام شود که شامل تقسیم ددقیقه بر 60 و ثانیه بر 3600 و تبدیل به زاویه نمودن آن است. تابع convert_coordinate همین کار را انجام نموده است که تصویرش در عکس زیر موجود است .

در تابع مربوط به پردازش داده مکانی، بعد از دریافت داده و انجام تبدیلات، urlمورد نیاز را آماده می کنیم که ساختار آن براساس فرمت لینک گوگل مپ می باشد. سپس دیتا آماده شده را به سمت مقصد مدنظر

می فرستیم که میان آن 3 ثانیه وقفه قرار میدهیم.

```
u32 ret = 0;
u8 result1[300] = "";
u8 result2[300] = "";
char *token;
u8 *parts[13];
u8 url[1000]="";
// if (stable == 1)
if (m_mqtt_state == STATE_MQTT_TOTAL_NUM && stable == 1)
{
    step++;
    Ql_strcpy(postMsg, "");
    Ql_strcat(postMsg, RMC_BUFFER);
    Ql_strcat(postMsg, "\0");
    int i = 0;
    token = strtok(postMsg, ",");
    // Split the string into three parts
    while (token != NULL && i < 13) {
        parts[i] = token;
        i++;
        token = strtok(NULL, ",");
    }
    double l1= Ql_atof((const char *)parts[3]);
    double  l2 =Ql_atof((const char *)parts[5]);
    l1 = convert_coordinate(l1);
    l2 = convert_coordinate(l2);
    char result12[20];
    char result22[20];

    Ql_snprintf(result12, 20, "%f", l1);
    Ql_snprintf(result22, 20, "%f", l2);

    Ql_strcpy(result1, parts[3]);
    Ql_strcpy(result2, parts[5]);
    Ql_strcpy(url, "https://www.google.com/maps/@");
    Ql_strcat(url, result12);
    Ql_strcat(url, ",");
    Ql_strcat(url,result22);
    Ql_strcat(url, ",18z");
    Ql_strcat(url, ",");
    Ql_strcat(url, "?entry=ttu");
    set1X();
    setCursor(0, 3);
    pub_message_id++; // The range is 0-65535. It will be 0 only when<qos>=0.
    ret = RIL_MQTT_QMTPUB(connect_id, pub_message_id, QOS1_AT_LEASET_ONCE, 0, pu_topic, Ql_strlen(url), url);
    if (RIL_AT_SUCCESS == ret)
    {
        APP_DEBUG("//<Start publish a message to server\r\n");
    }
    else
    {
        APP_DEBUG("//<Publish a message to server failure,ret = %d\r\n", ret);
    }
}
Ql_Sleep(3000);
```

```
double convert_coordinate(double coordinate) {
    int integerPart = (int)(coordinate / 100);
    int decimalPart = (int)coordinate % 100;

    int degree = integerPart * 60;

    char str_decimalPart[20];
    Ql_snprintf(str_decimalPart, sizeof(str_decimalPart), "%d", decimalPart);
    int power = 6 - (int)Ql_strlen(str_decimalPart);

    int multipliedDecimalPart = decimalPart * (int)pow(10, power);

    int minutes = multipliedDecimalPart / 10000;
    minutes += degree;

    return (double)minutes / 60.0;
}
```

بخش دوم :mqtt

در ابتدا باید یک بروکر انتخاب کرده و آن را در فایل کد خود ست کنیم ما در این پروژه از mosquito به عنوان بروکر استفاده کرده و آن را ست می کنیم.

در پروژه نیاز به یک client id داشته و آن را به صورت رندوم ست می کنیم.

در بخش بعد نیاز به ست کرد یک تاپیک یونیک داریم که در ان packet ها فرستاده شود. که در اینجا گیرنده ما mqttx یه اپ است . که در ان جا هم این موارد ثبت شده.

تاپیک ما در اینجا : v1/devices/me/telemetry3 است .

```
//connect info
Enum_ConnectID connect_id = ConnectID_0;
u8 clientID[] = "mc60tkjtkitk\0";
u8 username[] = "";
u8 passwd[] = "";
//topic and data
u32 pub_message_id = 0;
u32 sub_message_id = 0;
static u8 test_data[128] = "{\"location\":3}\0";            //packet data
static u8 pu_topic[128] = "v1/devices/me/telemetry3\0";  //Publisher  topic
static u8 su_topic[128] = "v1/devices/me/attributes\0"; //Subscriber topic
```

همچنین host و پورت مربوطه ثبت می شود. که مربوط به بروکر ما است.

```
#define HOST_NAME "test.mosquitto.org"
#define HOST_PORT 1883
```

در حالت های زیر به مواردی همچون کانکت شدن , سابمیت کردن و پابلیش آن, اشاره دارد که با توجه به حالت ما یکی از موارد اجرا میشود

```
case URC_MQTT_OPEN:
{
    mqtt_urc_param_ptr = msg.param2;
    if (0 == mqtt_urc_param_ptr->result)
    {
        APP_DEBUG("//<Open a MQTT client successfully\r\n");
        m_mqtt_state = STATE_MQTT_CONN;
    }
    else
    {
        APP_DEBUG("//<Open a MQTT client failure,error = %d\r\n", mqtt_urc_param_ptr->result);
    }
```

```
case URC_MQTT_CONN:
{
    mqtt_urc_param_ptr = msg.param2;
    if (0 == mqtt_urc_param_ptr->result)
    {
        APP_DEBUG("//<Connect to MQTT server successfully\r\n");
        m_mqtt_state = STATE_MQTT_SUB;
    }
    else
    {
        APP_DEBUG("//<Connect to MQTT server failure,error = %d\r\n", mqtt_urc_param_ptr->result);
    }
```

```
case URC_MQTT_SUB:
{
    mqtt_urc_param_ptr = msg.param2;
    if ((0 == mqtt_urc_param_ptr->result) && (128 != mqtt_urc_param_ptr->sub_value[0]))
    {
        APP_DEBUG("//<Subscribe topics successfully\r\n");
        m_mqtt_state = STATE_MQTT_PUB;
    }
    else
    {
        APP_DEBUG("//<Subscribe topics failure,error = %d\r\n", mqtt_urc_param_ptr->result);
    }
}
```

```
case URC_MQTT_PUB:
{
    mqtt_urc_param_ptr = msg.param2;
    if (0 == mqtt_urc_param_ptr->result)
    {
        APP_DEBUG("//<Publish messages to MQTT server successfully\r\n");
        m_mqtt_state = STATE_MQTT_TOTAL_NUM;
    }
    else
    {
        APP_DEBUG("//<Publish messages to MQTT server failure,error = %d\r\n", mqtt_urc_param_ptr->result);
    }
}
```

در این قسمت به تولید url برای map و ارسال آن به بروکر می پردازیم تا کانسیوم شود.

که در ابتدا به جداسازی داده مکانی با ‘,’ و سپس تبدیل و ارسال آن به بروکر کار ما به پایان می رید.

Www.google.com/maps/@52.72,31.41,18z

مثلا url ارسالی به سمت بروکر برای استفاده است.

```
Ql_strcpy(result1, parts[3]);
Ql_strcpy(result2, parts[5]);
Ql_strcpy(url, "https://www.google.com/maps/@");
Ql_strcat(url, result12);
Ql_strcat(url, ",");
Ql_strcat(url,result22);
Ql_strcat(url, ",18z");
Ql_strcat(url, ",");
Ql_strcat(url, "?entry=ttu");
set1X();
setCursor(0, 3);
pub_message_id++; // The range is 0-65535. It will be 0 only when<qos>=0.
ret = RIL_MQTT_QMTPUB(connect_id, pub_message_id, QOS1_AT_LEASET_ONCE, 0, pu_topic, Ql_strlen(url), url);
if (RIL_AT_SUCCESS == ret)
{
```