Republic of Turkey

Kocaeli University

Faculty of Engineering

Electronic & Communication Eng.

Micro-Controllers Project

Wi-Fi ThingSpeak - Updating System using MSP430G2553 Microprocessor, LDR Sensor, Temperature Sensor (LM35) and ESP8266 Wi-Fi Module.

Submitted by

Hammam SAWALMA- 140207104

Supervised by

Prof. Dr. Oguzhan Urhan

Kocaeli, May 2018

Wi-Fi ThingSpeak - Updating System using MSP430G2553 microprocessor, LDR sensor, Temperature Sensor (LM35) and ESP8266 Wi-Fi Module.

Hammam SAWALMA - 140207104

Electronic & Communication Engineering, Kocaeli University

Relevant Github Page: https://github.com/hammamsawalma/MSP430WIFI

1. Abstract

An embedded system that measures the temperature of the atmosphere using **LM35** Temperature Sensor, and also detect the Light-Level in the atmosphere using **LDR** (Light Dependent Resistor), developed using the **MSP430G2553** microprocessor, and **ESP8266** Wi-Fi module.

The Data of the sensors and historical statistics are visualized on a live-updating cloud server (ThingSpeak).

2. Background

2.1 MSP430G2553

Texas Instruments MSP430G2553 is part of the MSP430 family of ultra-low-power microcontrollers featuring different sets of peripherals. The MSP430G2553 mixed signal microcontroller features a 16-bit RISC CPU, 16-bit registers, 16KB non-volatile memory, 512 bytes RAM, 10-bit ADC, UART, and two 16-bit timers. The G2x53 series is popular due to the low price and various package sizes. The MSP430G2553 acts as the master node (central hub) of this system– connecting the sensors and ESP8266.

2.2 ESP8266

ESP8266 WiFi module is a self-contained SoC with integrated TCP/IP protocol. The module is an extremely low-cost all-in-one solution for Wi-Fi connectivity for microcontrollers. The ESP8266 contains an API- Application Programming Interface (AT command set firmware) which allows any microcontroller to control the module using UART protocol. The ESP8266 is used in this project as a client to initiate TCP connections to a cloud server and transmit information via HTTP requests.

2.3 LM35 (Temperature Sensor)

The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range.

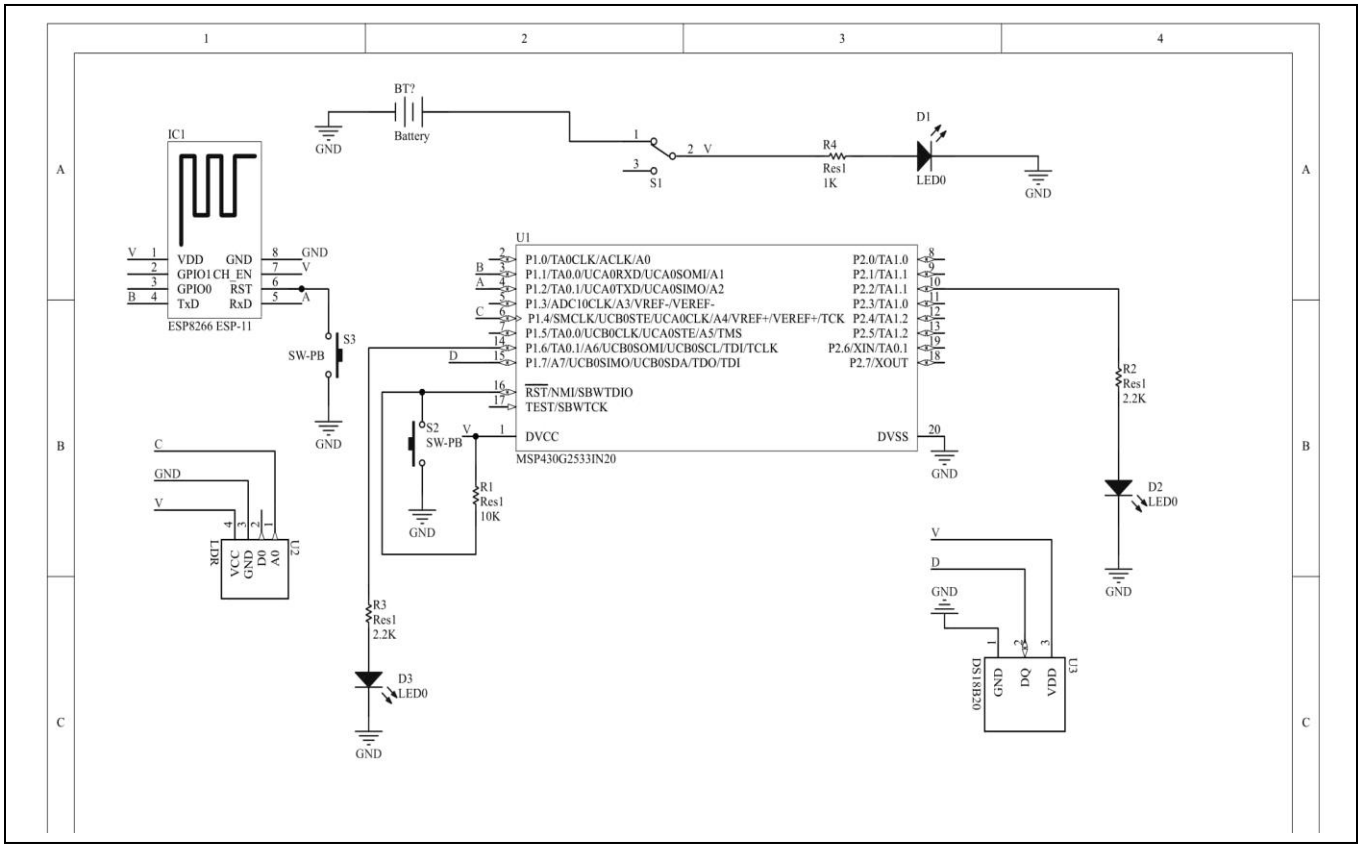
2.4 LDR Module (Light Dependent Resistor)

An LDR is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits.

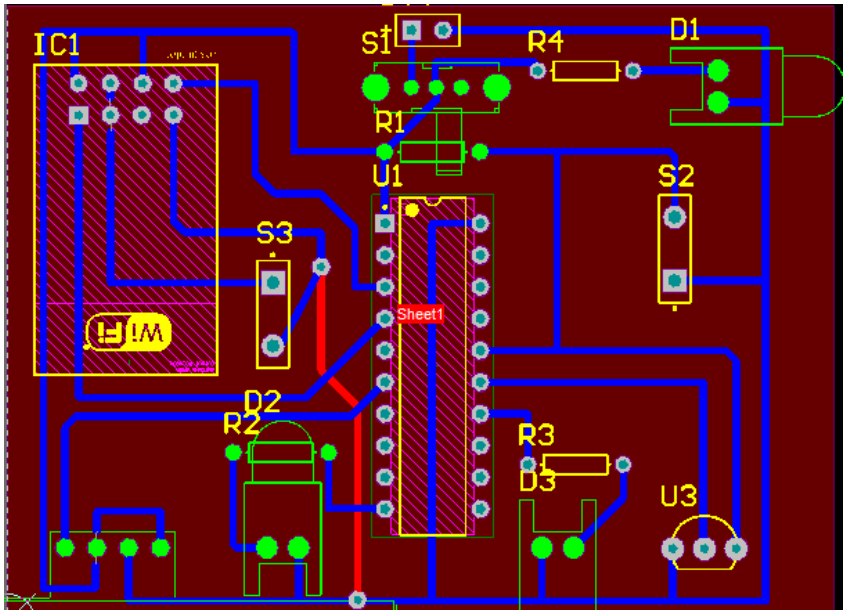
3. System Design

3.1 Schematic and PCB

This system's schematic was designed using Altium Designer 17.0 and can be seen in the Figure below... The circuit interfaces the MSP430 microprocessor with both the ESP8266 and the two sensors. The circuit provides several second-order design considerations such as resistors, and LED status indicators.



The PCB was designed for the system using Altium Designer 17.0. A 3D visualized render of the PCB is seen in Figure below:



3.2 MSP430G2553 Software

3.2.1 ADC Initialization (For both sensors)

The ADC ISR solves the distance equation using the ADC input. The MSP430 ADC registers are configured to the following parameters.

ADC Register Initializations

Register	Bit	Description
ADC10CTL0	REFON	Reference generator: ON
ADC10CTL0	SREF_0	Select reference VR+ = AVCC and VR- = AVSS
ADC10CTL0	ADC10ON	ADC10 Enable: ON
ADC10CTL0	ADC10IE	ADC10 Interrupt: ENABLED
ADC10CTL1	INCH_4	Input channel A4 (P1.4) – LDR Sensor
ADC10CTL1	INCH_7	Input channel A7 (P1.7) – TMP Sensor

ADC10AE0	BIT4	ADC Analog enable A4- LDR Sensor
ADC10AE0	BIT7	ADC Analog enable A7- TMP Sensor

For LDR:

The levels of the light was set due to the ADC10MEM 10-bit register (Max. 1023) where 5 indicates a very high light and 1 indicates a very low light.

```
// LDR Levels ( 5 --> Light is so high , 1 --> Light is so low )
    LDR = 0;
    if ( LDR_Result > 0 && LDR_Result < 100 ) { LDR = 5 ; }
    else if ( LDR_Result > 100 && LDR_Result < 250 ) { LDR = 4 ; }
    else if ( LDR_Result > 250 && LDR_Result < 450 ) { LDR = 3 ; }
    else if ( LDR_Result > 450 && LDR_Result < 650 ) { LDR = 2 ; }
    else if ( 650 < LDR_Result ) { LDR = 1 ;}
```

3.2.2 UART Initialization

UART is used to communicate (RX/TX) with the ESP8266 WiFi module. The ESP8266 communicates at 115200 baud with about 3.3V logic.

UART Register Initializations

Register	Bit	Description
DCOCTL	0	Select lowest DCOx and MODx settings
DCOCTL	CALDCO_1MHZ	Set DCO
BCSCTL1	CALBC1_1MHZ	Set DCO
P1SEL	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
P1SEL2	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
UCA0CTL1	UCSSEL_2	Set UART use SMCLK
UCA0BR0	8	1MHz 115200 baud
UCA0BR1	0	1MHz 115200 baud
UCA0MCTL	UCBRS2	Modulation UCBRSx = 5
UCA0MCTL	UCBRS0	Modulation UCBRSx = 5

UCA0CTL1	~UCSWRST	Initialize USCI state machine

Functions are created to send UART strings to/from the ESP8266.

Function	**Description**
void TX(**const** char *s)	Send a string via UART
void putc(**const** unsigned c)	Send a character via UART
void crlf(void)	Sends a carriage return and new line via UART

3.2.3 Timer Initialization

A timer is initialized to send distance data to the web server on a specified interval. The MSP430G2553 contains a 1MHz SMCLK. The maximum clock divider available is 8-times. Because the MSP430G2553 uses 16-bit registers the maximum period is 65535. Therefore, the slowest hardware frequency for a timer is 2Hz.

$$2\text{Hz} = \frac{(1\text{MHz})/8}{\text{Period}}$$

Solving for the timer period for a 2Hz interval results in a period of 62500. This is the maximum value before we reach an overflow of 65535.

Timer A Register Initializations

Register	**Bit**	**Description**
TA0CCTL0	CCIE	Enable interrupts for Timer A0
TA0CTL	TASSEL_2	Select SMCLK clock source
TA0CTL	MC_1	Set timer mode to UPMODE
TA0CTL	ID_3	Set clock divider to SMCLK/8
TA0CCR0	62500	Set capture compare register period to 62500

Using software, the timer can be acted upon in slower intervals. Because we know the timer interrupts every 2Hz (0.5 Second) we can use software to determine when to activate code. By the way, because of the “Timer Interrupt Processing Time -which contain some delay cycles- it exceeds the 2Hz (0.5 Second),

it is about 25 second, so there will always be a pending interrupt so the code will act like there is a loop inside the timer interrupt function.

3.2.4 WiFi Functions

The program has functions for sending specific commands to the ESP8266 to send and format data for upload. The program requires definitions for Wi-Fi information such as SSID, Password, URL, etc.
The program requires definitions for WiFi information such as SSID, Password, URL, etc.

WiFi Constant Definitions

Constant	**Description**
MAX_SEND_LENGTH	Determines maximum number of digits of sensor data to send (default 3)
BASE_URL_LDR	The URL for API request servicer – LDR sensor
BASE_URL_TMP	The URL for API request servicer – TMP sensor
TCP_REQUEST	Preformatted message to initiate TCP request
WIFI_NETWORK_SSID	SSID for WiFi network
WIFI_NETWORK_PASS	Password for WiFi network
ENABLE_CONFIG_WIFI	Enable to configure the WiFi network to settings

Functions are created perform WiFi related tasks.

Function	**Description**
void WifiConfigureNetwork(void)	Configure the network to defined settings
void WiFiSendTCP()	Initiate TCP request with web server
void WiFiSendLength(unsigned int data)	Send the length of the incoming TCP request to ESP8266 before upload- Was set as 50- Useless function

void WiFiSendMessage(unsigned int data)	Send the message with data to web server for upload

4. System Performance