# HARDWARE CONNECTIVITY AND SOFTWARE DEVELOPMENT FOR HIGH ALTITUDE BALLOON WIRELESS DATA ACQUISITION SYSTEM

**Vidya Gopalakrishnan (A20249496)**

**Department of Electrical Engineering**
**Illinois Institute of Technology**

***Abstract –*** *The communication system described in this paper is used to facilitate the easy data extraction of the payload inputs wirelessly and in real time for high altitude balloon launches. This report gives detailed hardware description and software development planning with flowcharts .The connectivity to the microcontroller to each of the equipment has also been described .Last but not the least, a brief glimpse to the testing phase has also been mentioned. This project when completed will be for the high altitude balloon launches for Adler Planetarium*

## 1. INTRODUCTION

This project is the second part of the requirement for Adler Planetarium. The Far Horizons project is the back end part of planetarium which organises high altitude balloon launches for students , teachers and amateur hobbyists ,that send simple experiments into near space .To recap ,the first part of the project had two requirements , first one being controlling cut down system using the distance calculated by the GPS tracker. This mainly aimed to control the amount of distance the balloon goes so that it stays within a 50 mile radius irrespective of the height it would have reached. Secondly, to design a communication system for real data acquisition from the balloon. In the first part of the project the communication system was designed and the link budget calculated. The transceiver system was also decided during the first part of the project.

This paper is the second part of the project which aims at detailed study of the hardware used such as the microcontroller, the transceiver and the some inputs . Secondly the entire connectivity of the system. Last but not the least the software development. This paper also puts the starting stages of testing the system as one communication unit.

## 2. HARDWARE DESCRIPTION & BLOCK DIAGRAMS

Two possible inputs were considered for the process of testing the transmission and reception using the wireless transceivers. One is from the Temperature Sensor IC chip and the second is from RS323 TTL camera. Images /Data will be first stored in the SD card using a microcontroller and then from the SD card using the microcontroller these data/images will transmit using the microcontroller to the receiver .The receiver itself will be microcontroller which will in turn be connected to an SD card interface. Data Received will be stored in this SD card and then displayed on LCD/monitor. A simple block diagram of the transmitter side is shown below:

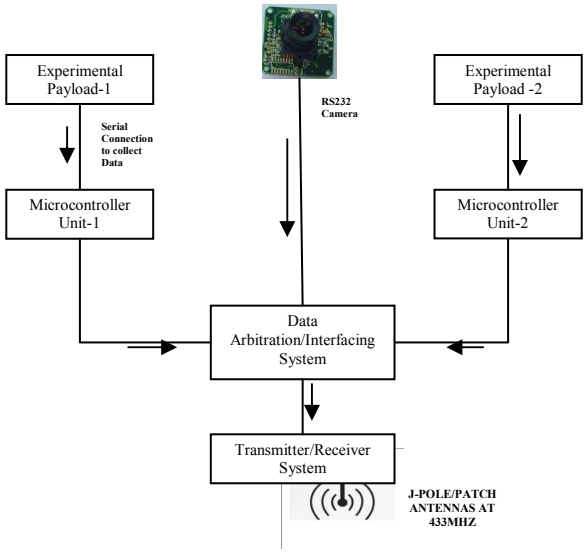



**Fig 2.1 Communication System Block Diagram – Transmitter Side**

The block diagram is the part of the receiver side:

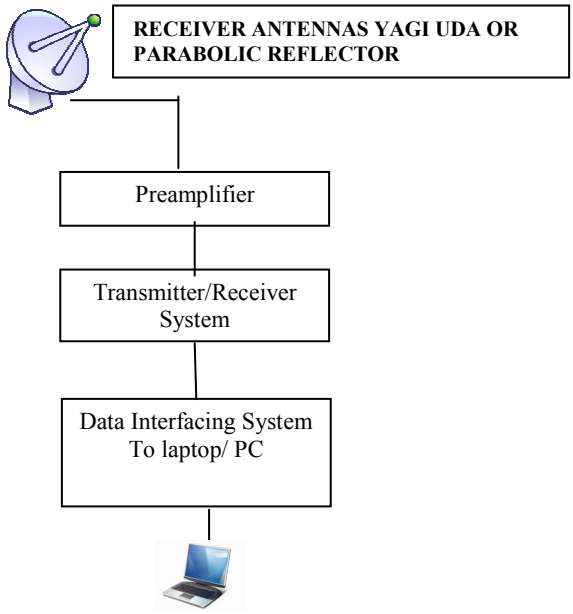



**Fig 2.2 Communication System Block Diagram – Receiver side**

A detailed block diagram with pin out and connectivity has been shown in Appendix A.

### 2.1 Temperature Sensor [1]

Temperature sensor is one of the inputs used to test the transmit/receive process. The IC chip used for this purpose is This IC can measure temperatures from -55 $^{o}$C to +125$^{o}$C .The thermostatic settings can be defined by the user and they are non-volatile.and the temperature is read as 9 bit value .Data can be read from and written to the temperature sensor using 3 wire interface which is the CLK DQ and the RST button and it converts the data read into digital word in about 750 ms .

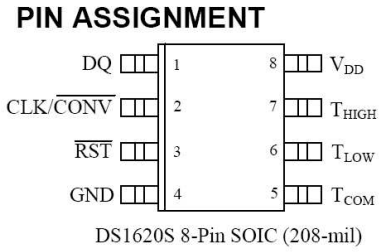
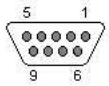



**Fig 2.1.1 Pin Diagram for Temperature Sensor [1]**

### 2.2 RS232 TTL Camera [5]

**Fig 2.2.1 RS232 TTL CAMERA[5]**

The camera allows to capture Jpeg images and transfer it using the serial port to interface to the PC/ SD card for storing .These cameras require a 5V power supply and provides JPEG image with a resolution of VGA/QVGA/ 160 X 120.It also has the capability to run live video that is captured by the camera. The camera has its own GUI which can be used to control the images/video captured if a microcontroller is not used .The baud rate of this camera is 38400.The pin configuration of this camera is 5 pins . One pin is for TV output which can be used only if necessary. What is important for this project is the Rx (input) pin, Tx (output) pin, GND and 5V.The connectivity to Arduino is given in the following sections. A DB9 serial interface can also be used to connect the camera to the PC or a microcontroller. The DB9 pin outs are as shown below:




9 PIN DE-9 FEMALE at the Cable.

| Pin | Name | RS232 | V.24 | Dir | Description |
|---|---|---|---|---|---|
| 1 | (D)CD | CF | 109 | ← | (Data) Carrier Detect |
| 2 | RXD | BB | 104 | ← | Receive Data |
| 3 | TXD | BA | 103 | → | Transmit Data |
| 4 | DTR | CD | 108.2 | → | Data Terminal Ready |
| 5 | GND | AB | 102 | — | System Ground |
| 6 | DSR | CC | 107 | ← | Data Set Ready |
| 7 | RTS | CA | 105 | → | Request to Send |
| 8 | CTS | CB | 106 | ← | Clear to Send |
| 9 | RI | CE | 125 | ← | Ring Indicator |

**Fig 2.2.2 DB9 pin out for TTL camera[5]**

### 2.3 SD Card Interface [11]

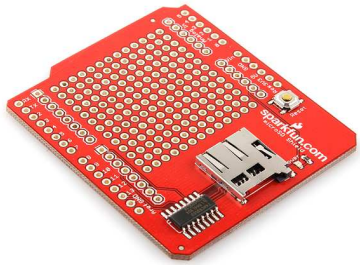


**Fig 2.3.1 SD Card Interface [11]**

The microcontroller used for this project does not have enough memory to store data from any retrieved input device. Hence a SD card shield interface is used to transfer the data from the input devices to the SD card using the microcontroller as the programming device.

The micro SD shield acts as an interface that can be used to connect the SD card to the microcontroller .This can be used to store data into the SD card as well as retrieved data from the SD card which is what is required for this project .The interface is connected to the microcontroller using the SPI interface .SPI interface is the Serial Peripheral Interface

The pin out diagram of the SD card interface is as shown below. The connectivity to the board for MCU will be explained in detail in the following sections

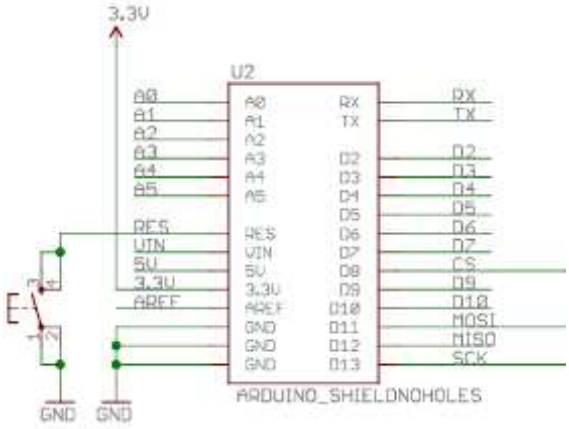



Fig 2.3.2 Pin Diagram for SD Card Interface

## 2.4 Micro Controller

There are two microcontrollers which are used for this project which are both from the same board manufacturer called Arduino. The Atmega microcontrollers are used for these boards. It is basically an open source electronics used by hobbyists, designers, artists etc to creative interactive objects or environment. It has its own coding methodology which is comparable C/C++. A number of libraries to assist the users which includes SPI library to control interfaces needing there SPI interface.

### 2.4.1 Arduino Uno:[10]

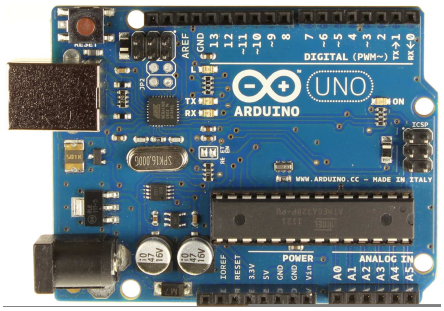



Fig 2.4.1 Arduino Uno [10]

Arduino Uno uses the Atmega 328 as its microcontroller. The board is made with 14 digital output pins. This includes 6 pins that can be used for PWM outputs and 6 pins that can be used for analog outputs .It also have a 16MHz crystal oscillator. There is a power jack for external power source as well a USB connection for programming as well as power connection. The reset button as usual can be used to restart the program when eve required .The operating voltage of this microcontroller board is about 5V .It has a flash memory of 32kb,a SRAM of 2kB . The EEPROM is of 1kB with a clock speed of 16MHz..

### 2.4.2 Arduino Mega:[9]

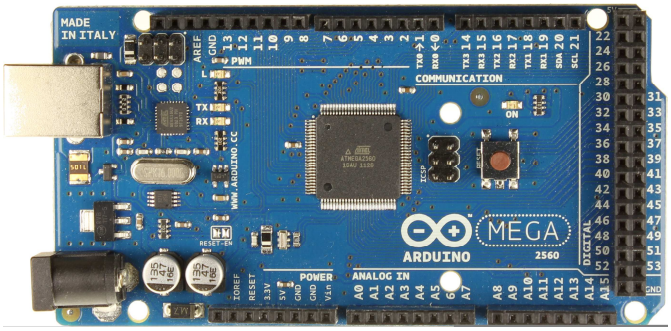



Fig 2.4.2 Arduino Mega 2560[9]

Arduino Mega 2560 is used for this project as the other microcontroller board. This board uses the Atmega 2560 as its microcontroller hence the name for the board. The board is made with 54 digital input/output pins. This includes 14 pins that can be used for PWM outputs and 16 pins that can be used for analog outputs .It also have a 16MHz crystal oscillator. There is a power jack for external power source as well a USB connection for programming as well as power connection. The reset button as usual can be used to restart the program when eve required .The operating voltage of this microcontroller board is about 5V .It has a flash memory of 256kB of which 8kB is used for the boot loader ,a SRAM of 8kB . The EEPROM is of 4kB with a clock speed of same as that of Arduino Uno microcontroller.

## 2.5 Transceiver [12]

The transceiver used a CC1101 433 MHZ frequency from Texas Instruments.

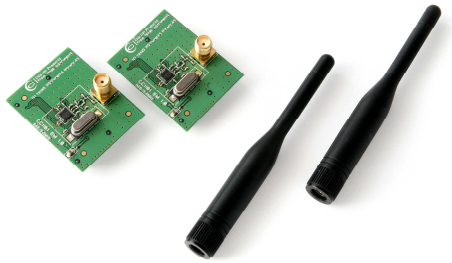


Fig 2.5.1 CC1101 EMK – Evaluation module [12]

The data transfer characteristics of the transceiver at 433MHz is as shown below:

- **0.6 k Baud data rate,** Receiver sensitivity -116 dbm
- **1.2 k Baud data rate,** Receiver Sensitivity -112dbM

- **38.4 k Baud data rate,** Receiver Sensitivity -104dbm
- **250 k Baud data rate,** Receiver sensitivity -95 dbm

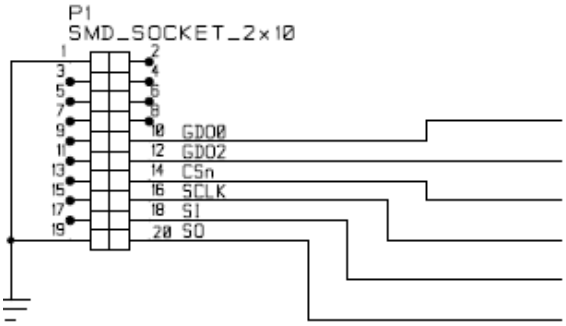



**Fig 2.5.2 CC1101 EMK – Pinout Diagram [12]**

The transmitter and the receiver are connected to the Arduino using the SPI interface links. Two other pins that are important for the functioning of the CC1101 pins GDOxx .The SPI interface pins are as follows:

**GDO0** – as a trigger to send or receive data. It is set to one while the packets are being sent or received and then set back to zero while the process is completed.

**GDO2** – this is the serial clock output

## 3. SOFTWARE DEVELOPMENT PLAN

The first of two parts of the development plan is for the Data Transfer side of the High Altitude Balloon Wireless Communication Module which is the transmitter side. The flowcharts for the software can be found in Appendix C.

### 3.1 Connectivity Between Input Devices and Microcontroller/Storing Data in SD card-Part 1:

**Module 1:** Programming RS232 TTL Camera to take pictures using Arduino Mega2560

**Module 2:** Programming Arduino Mega2560 to store the pictures taken by the camera in an SD card. SD card can be connected to an SD card Module which comes from Arduino.

**Plan of Action:** Arduino Mega 2560 comes with a tin built library for RS232 TTL camera .This will allow the capture of pictures and video frames .This MCU also comes with libraries for writing to and reading from an SD camera. The SD card sub-module can be easily configured using these libraries .The sample program with the MCU gives a detailed description of how to code the camera to take pictures and transfer it to an SD.

The SD card module can be connected to the MCU using an SPI interface. This is a Serial Peripheral Interface which talks to the MCU using four wires .The fours wires are MOSI( Master Out Slave In) MISO(Master In Slave Out) SCLK (Serial Clock ) and SS(Slave Select) . SS can be used if there is more than one module connected to the MCU using SPI. For example, in case of this current project both the SD card module as well as the CC1101 can be connected to the same SPI pins using the SS pin to select between the two modules.

### 3.3 Connectivity between Transceiver and Microcontroller /Transferring Data to Register of CC1101- Part 2:

**Module 1:** Programming Arduino Mega2560 to talk to CC1101 using SPI interface to transfer data

**Modules 2:** Programming MCU to transfer data between the SD card and the CC1101 transceiver at 15 kbps per second. The SD card will act as a buffer to store and transfer the data.

**Plan of Action:** Programming for Arduino is based on C/C++ coding methodology .Libraries is also available for the same along with an Example code.

### 3.2 Transferring Data between two transceivers wirelessly-Part 3:

**Module 1:** Programming the CC1101 to transfer the Data to transmit using MCU and also to receive any control commands from the receiver side.

**Plan of Action:** CC1101 can also be programmed to be controlled by the MCU .The modulation, bit rate, transmit mode etc can be detailed in the program for MCU and used to transfer images as per requirement. The CC1101 should also be programmed to receiver control signals if possible from the receiver so as to know when to begin transmitting the data as well as ACK signals

The Receiver side will be similar to that of the transmitter side except it will be the reverse process. The Receiver side process will be divided into two parts

### 3.3 Collecting transferred data from RXFIFO to SD Card-Part 4:

**Module 1:** Programming CC1101using MCU to receive data from the transmitter to receiver.

**Module 2:** Storing data received from the receiver to the SD card.

### 3.4 Displaying Data from SD card to Serial Port / LCD - Part 5:

**Module 1:** Programming the MCU to transfer data from SD card to PC Monitor/LCD Monitor.

## 4. CONNECTIVITY DETAILS TO MICROCONTROLLER

The pin out descriptions for each of these equipments is shown above in the hardware description section

### 4.1 Camera Connectivity: [4]

a).Arduino pin 2 to Camera TX
b).Arduino pin 3 to Camera RX
c).Arduino 5V VCC to Camera VCC
d).Arduino GND to Camera GND

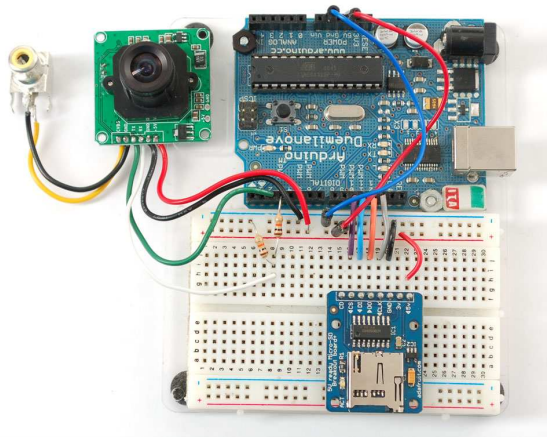


**Fig 4.1.1 Camera Connectivity to Arduino Uno/Mega2560**

This connectivity is the same for Arduino Uno and Arduino Mega 2560.

### 4.2 Temperature Sensor Connectivity to microcontroller:[5]

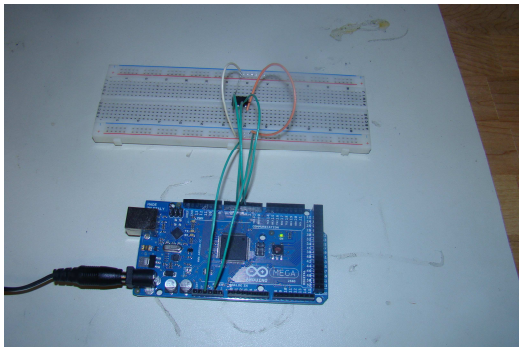


**Fig 4.2.1 Temperature Sensor to Arduino Uno/Meg2560**

a).Arduino pin 3 to DS1620 RST
b).Arduino pin 4 to DS1620 CLK
c).Arduino pin 5 to DS1620 DQ
d).Arduino 5V to DS1620 VDD
e).Arduino GND to DS1620 GND

This connectivity is the same for Arduino Uno and Arduino Mega 2560

### 4.3 SD Card Interface Connectivity:

These pin connections are for Arduino Mega 2560 to SD Card Interface:

a).Arduino pin 53(Csn/SS) to SD Shield pin D10
b).Arduino pin 52(MOSI) to SD Shield pin D11
c).Arduino pin 50(MISO) to SD Shield pin D12

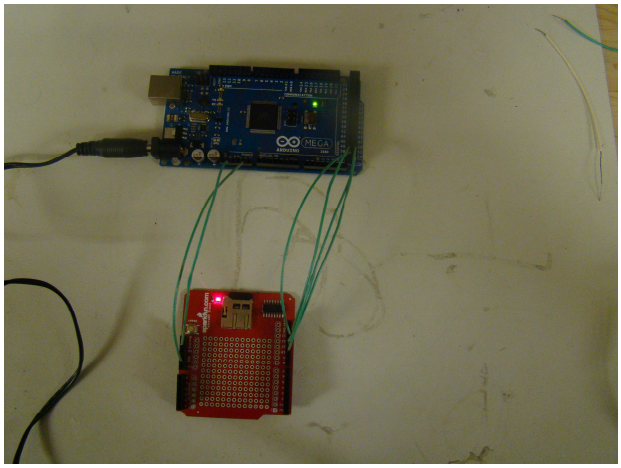


**Fig 4.3.1 SD Card Interface to Arduino Mega2560**

d).Arduino pin 51 (CLK) to SD Shield pin D13
e).Arduino pin 3.3V to SD Shield 3.3V
f).Arduino pin GND to SD Shield pin GND

In the case of Arduino Uno the pin numbers that are assigned for the SPI interface will be different:

a).Arduino pin 10 (Csn /SS) to SD Shield pin D10
b).Arduino pin 11(MOSI) to SD Shield pin D11
c).Arduino pin 12(MISO) to SD Shield pin D12
d).Arduino pin 13 (CLK) to SD Shield pin D13
e).Arduino pin 3.3V to SD Shield 3.3V
f).Arduino pin GND to SD Shield pin GND

### 4.4 Transceiver Connectivity:

CC1101 consists of two sockets which consists P1 and P2 as explained in the hardware description. The CC1101 will also be connected to the SPI interface of the Arduino microcontrollers hence there will be two slaves being controlled by the microcontroller which are then controlled alternatively by the MCU. The connectivity required is same as that of the SPI interface. A screen shot of the connectivity is shown below.

Transmitter side requiring better processing than the receiver side Arduino Mega 2560 is used for this purpose.

a).Arduino pin 53 (Csn /SS) to CC1101 pin 14
b).Arduino pin 50 (MOSI) to CC1101 pin 18
c).Arduino pin 51 (MISO) to CC1101 pin 20

d).Arduino pin 52 (CLK) to CC1101 pin 16
e).Arduino pin 3.3V to SD Shield 3.3V
f).Arduino pin GND to SD Shield pin GND
g).Arduino pin 2 to CC1101 pin 10
h).Arduino pin 9 to CC1101 pin 12

Arduino Uno is being used to control the receiver side as shown in the picture below

a).Arduino pin 10 (Csn /SS) to CC1101 pin 14
b).Arduino pin 11(MOSI) to CC1101 pin 18
c).Arduino pin 12(MISO) to CC1101 pin 20
d).Arduino pin 13 (CLK) to CC1101 pin 16
e).Arduino pin 3.3V to SD Shield 3.3V
f).Arduino pin GND to SD Shield pin GND
g).Arduino pin 2 to CC1101 pin 10
h).Arduino pin 9 to CC1101 pin 12

## 5. LIBRARIES PROVIDED BY ARDUINO [6]

A number of libraries are provided by Arduino as well as the other amateur system designers to support the functionalities required for the projects . Some of the libraries that have been used for this project are shown below.

**SoftSerial Library:** Generally in Arduino the TX/RX pins in the case of Mega 2560 are pin0,pin1,pin14, pin15, pin16, pin17, pin18, pin19.But using this Serial Software Library from Arduino any pins in this MCU can be used for Serial communication .This library helps to virtually simulate these pins. Some issues as this are a code that is not actually supported by the hardware is that:

- Only speeds up to 9600 baud work
- Serial.available () doesn't work
- Serial.read() will wait until data arrives
- Only data received while Serial. Read () is being called will be received. Data received at other times will be lost, since the chip is not "listening".

**SD Library:** This is basically used to provide the necessary read write commands for connecting an SD card breakout shield to the Arduino pin. As shown above the SPI interface of the Arduino is used to connect to these SD shields. It is important to format the card before starting to use it as this library as well as Arduino supports only FAT 16 and FAT 32 formatted card.

**ELECHOUSE CC1101 Library:** This is basically used to provide the necessary read write commands for connecting the CC1101EMK to the Arduino. These have function from copying data into the buffers to transmitting the data between the transmitter and the receiver. Some of the functions that are included in this library include Initliazing CC1101, Send Data, Receive Data, Setup transceiver to receiver data, Initilializing input and output pins etc.

**SPI Library (Serial Peripheral Interface):**

As the name suggests this library is used to control and connect device s that can only e connected using SPI ports . Typically there are three lines common to all the devices,

Master In Slave Out (MISO) - The Slave line for sending data to the master,

Master Out Slave In (MOSI) - The Master line for sending data to the peripherals,

Serial Clock (SCK) - The clock pulses which synchronize data transmission generated by the master, and

Slave Select pin - the pin on each device that the master can

Communication between the Master and the Slave takes place when the Slave Select pin(SS) is low
SPI allows three modes of transmission , hence this library is also written to call functions that allow these modes. According, to the data transmission modes set , the data can be shifted in and out on the rising edge of the clock or the falling edge of the clock . The three modes are decided by setting the clock signal a clock priority high or low .

## 6. TEST 2 & RESULT 2 - TRANSFERRING SIMPLE DATA BETWEEN TRANSMITTER AND RECEIVER

This test was checked to see if the data can be transmitted from an input device to the SD card

### 6.1 Test:

First and foremost according to the project plan is to transfer the data from an input device to a storage device .The input device used for this purpose was a temperature sensor and the storage device was a 4GB SD card. As explained earlier both the devices are connected to the microcontroller according to the pin connectivity show in previous section.

To give a short description, the temperature sensor is connected to the microcontroller according to the connectivity diagram. The program is executed. The program is shown in APPENDIX 1 .There are two parts of this program:

1). Reads the data from the temperature and displays it on the serial port screen on the laptop that the microcontroller is connected to

2). the data collected by the temperature sensor is at the same time stored in a file in the SD card for further analysis.

The output screen shots of the temperature sensor readings on the serial port to the screen of the laptop are shown below.

### 6.2 Result:

The data was successful displayed on the laptop screen using the serial port at the same time the data was also stored in the SD card. Output from the temperature sensor reader to the serial port can be found in Appendix B

## 7. TEST 2 & RESULT 2 - TRANSFERRING SIMPLE DATA BETWEEN TRANSMITTER AND RECEIVER

Before sending Data from an input device it was considered best to send simple numbers between the transmitter and the receiver wireless.

7.1 Test:

The simple program was created to send number 0 to 61 from the transmitter to the receiver. This program was a part of the library of files that was provided by ELECHOUSE for the CC1101 programming with arduino .

First and foremost , the library files had to be checked for appropriate register settings . This was done using a program tool GUI called the Smart RF. Connecting the transceiver Evaluation module to the Smart RF board which then be connected to the laptop using a USB cable . The registers in the CCC1101 can then be read . The parameters shown in Appendix B is mandatorily written into the registers for CC1101 for the transmit receive to work .

After making the necessary corrections, the program is uploaded on both the transmitter and receiver . The microcontrollers used on both the sides are explained in the Connectivity section .Arduino has loop function in the main program which keeps sending numbers from 0 to 60 to the receiver .

### 7.2 Result:

This test as of now is still being conducted .As for now the errors that is being speculated is the fact that the register settings required for the transmitter and receiver to communicate is not being read properly from the microcontroller through the SPI to the registers of the CC1101.Unfortunatley since this part is still under test there Is currently no output screen shot to be shown in the paper

## 8. FUTURE STEPS AND ASPIRATIONS

The following steps will be the plan followed for future developments in the project:

1).Successfully transmitting small values such readings from the temperature sensor wirelessly in the lab environment
2). Using RS232 camera to capture pictures and transfer to SD card in lab environment
3).Transmit the pictures wirelessly in lab environment between transmitter and receiver
4). Debugging possible synchronization and timing issues which can occur during the picture transfer
5). Connect multiple input devices and send each data one at a time by using SS pin in the SPI interface to choose the inputs to send data to SD card in lab environment
6). Antenna design, making and testing
7). Launching the communication system in real time for high altitude balloon launch to capture real time data.

## 9. CONCLUSION

This part of the Adler High altitude balloon real time data acquisition system revolved around hardware connectivity and software planning .The pin out details and connection details to each of the input and output components was found tried and tested . Programming and software development was conducted for the transferring data from and input device to an SD card which ran successful. Unfortunately it was not the same for in lab transmission and reception of data wirelessly . Further tests are being conducted and developments will be made to launch a successful and functioning live data acquisition system

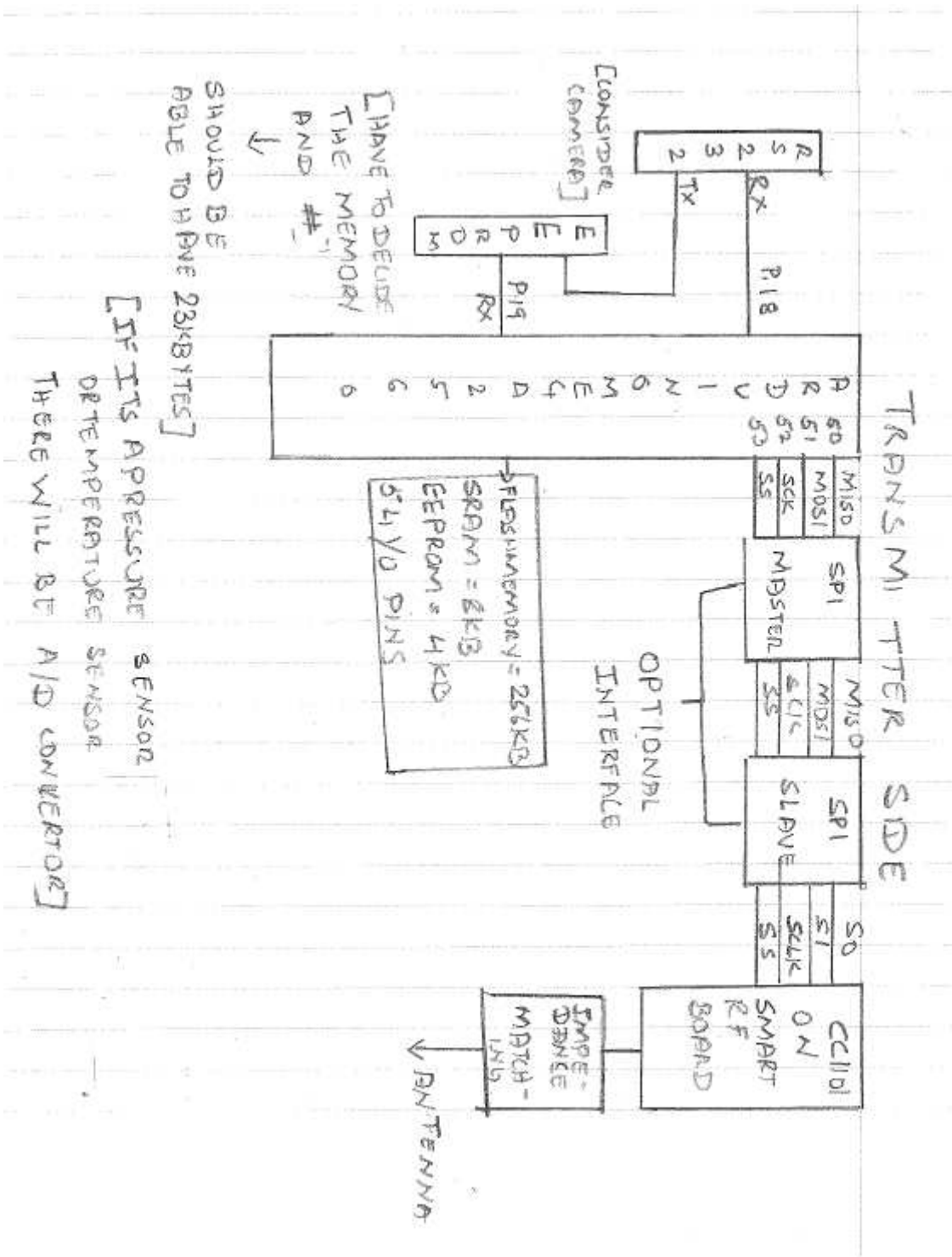
## 10. REFERENCES

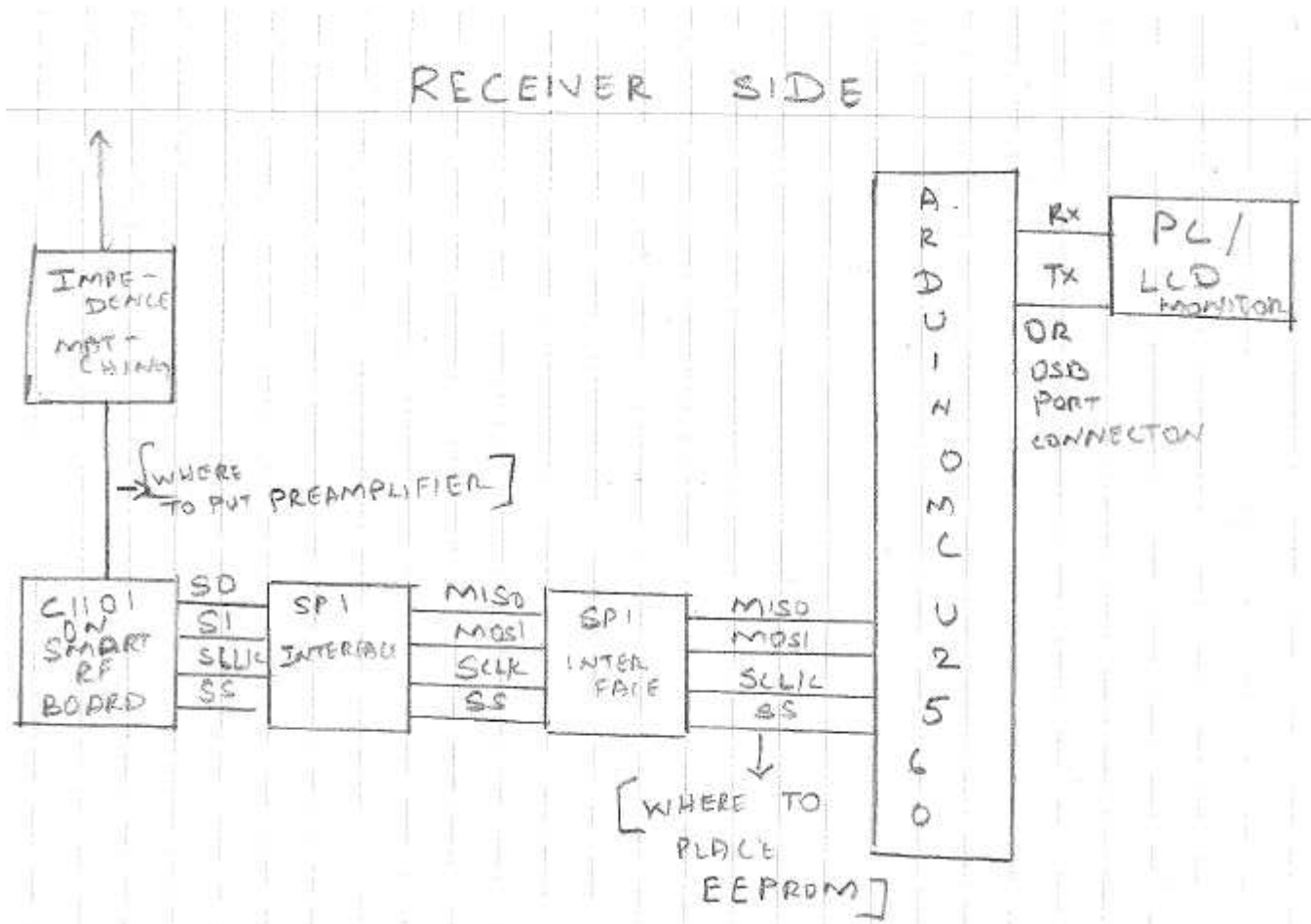

[1] Dallas Semiconductor Maxim DS1620 Digital Temperature Sensor Manual

[2].www.elexchouse.com

[3].Texas Instruments CC1101 Transceiver Datasheet
[4].http://rubenlaguna.com/wp/2008/08/31/arduino-and-ds1620-digital-temperature-sensor/
[5]. http://www.ladyada.net/products/camera/"
[6].Basics of Arduino- Programming and Instruction Book
[7]. CC1101 Elechouse Manual
[8].http://www.avrfreaks.net/index.php?name=PNphpBB2&file=printview&t=85576&start=0
[9]. http://www.arduino.cc/en/Main/ArduinoBoardMega2560
[10]. http://www.arduino.cc/en/Main/ArduinoBoardUno
[11]. http://www.sparkfun.com/products/9802
[12]. http://www.ti.com/tool/CC1101EM433_REFDES **[12]**
**[13]. CC1101EM_434MHz_SCHEMATIC_2_0_0.pdf**

# APPENDIX A- BLOCK DIAGRAM




RECEIVER SIDE
IMPE-
DENCE
MATT
CHING
[WHERE
TO PUT PREAMPLIFIER]
CC1101
ON
SMART
RF
BOARD
SO
SI
SCLK
SS
SPI
INTERFACE
MISO
MOSI
SCLK
SS
SPI
INTER
FACE
MISO
MOSI
SCLK
SS
[WHERE TO
PLACE
EEPROM]
ARDUINO MC U2560
Rx
Tx
PC/
LCD
MONITOR
OR
OSD
PORT
CONNECTON

# APPENDIX B – CODE FOR PROJECT

**Code for Transferring Temperature Sensor Data to the SD Card :**

```
/*
  Arduino    DS1620
  pin 3  ->    RST
  pin 4  ->    CLK
  pin 5  ->    DQ

   */

// include the SoftwareSerial library so you can use its functions:
#include <SoftwareSerial.h>
#include <SD.h>

#define rxPin 0
#define txPin 1
#define ledPin 13
#define buttonPin 2

#define rstPin 3
#define clkPin 4
#define dqPin 5
```

**//Setting up Slave Select Pin for SPI Interface**

```
const int chipSelect = 10;
```

**// set up a new serial port using Software Serial**

```
SoftwareSerial mySerial =  SoftwareSerial(rxPin, txPin);
byte pinState = 0;

void setup()  {
```

**// define pin modes for tx, rx, led pins:**

```
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(rstPin, OUTPUT);
  pinMode(clkPin, OUTPUT);
  pinMode(dqPin, OUTPUT);
  pinMode (10,OUTPUT);
```

**// set the data rate for the SoftwareSerial port**

```
  mySerial.begin(9600);
  mySerial.print("Initializing SD Card...");
```

**//Inirializing SD Card**

```
  if (!SD.begin(chipSelect))
```

```
  {
    mySerial.println("Card failed, or not present");
    return;
  }

  mySerial.println("card initialized.");

}

void loop()
{

    rst_low();

    clk_high();
    rst_high(); //all data transfer are initiated by driving RST high
    write_command(0x0c); // write config command
    write_command(0x02); // cpu mode
    rst_low();
    delay(200); //wait until the configuration register is written

    clk_high();
    rst_high();
    write_command(0xEE); //start conversion
    rst_low();
    delay(200);

    clk_high();
    rst_high();
    write_command(0xAA);
    int raw_data = read_raw_data();
    rst_low();

    mySerial.print("temperature:");
    mySerial.print(raw_data/2);
    mySerial.println(" C");
    delay(100);

    //toggle an LED just so you see the thing's alive.

  toggle(13);

  // Creating a Logging File on the SD Card

  File dataFile = SD.open("datalog.txt", FILE_WRITE);
 // Writing to the logging file
if (dataFile)
{
  dataFile.print("temperature=");
  dataFile.println(raw_data/2);
  dataFile.close();
}
else
{
  mySerial.println("error opening datalog.txt");
}
delay(500);

}
```

**//Functions Called during the Temperature Sensor Reading  Process by the Micorcontroller**

```
void toggle(int pinNum)
{
  // set the LED pin using the pinState variable:
  digitalWrite(pinNum, pinState);
  // if pinState = 0, set it to 1, and vice versa:
  pinState = !pinState;
}

void write_command(int command)
/* sends 8 bit command on DQ output, least sig bit first */
{
  int n, bit;

  for(n=0;n<8;n++)
  {
    bit = ((command >> n) & (0x01));
    out_bit(bit);
  }
}

int read_raw_data(void)
{
  int bit,n;
  int raw_data=0;

  pinMode(dqPin,INPUT);

     /* jam the dq lead high to use as input */
  for(n=0;n<9;n++)
  {
     clk_low();
     bit=(digitalRead(dqPin));
     clk_high();
     raw_data = raw_data | (bit <<  n);
  }
  pinMode(dqPin, OUTPUT);
  return(raw_data);
}

void out_bit(int bit)
{
  digitalWrite(dqPin, bit);  /* set up the data */
  clk_low();              /* and then provide a clock pulse */
  clk_high();
}

void clk_high(void)
{
  digitalWrite(clkPin,HIGH);
}

void clk_low(void)
{
  digitalWrite(clkPin,LOW);
}

void rst_high(void)
{
```

```
  digitalWrite(rstPin,HIGH);
}

void rst_low(void)
{
  digitalWrite(rstPin,LOW);
}
```

**Code for Transmitting simple numbers from 0 to 60:**

**Transmitter Side :**

```
#include <ELECHOUSE_CC1101_2560.h>

#define size 61

byte TX_buffer[size]={0};
byte i;

void setup()
{
 Serial.begin(9600);
 ELECHOUSE_cc1101.Init();
 for(i=0;i<size;i++)
 {
   TX_buffer[i]=i;
 }
}

void loop()
{
 ELECHOUSE_cc1101.SendData(TX_buffer,size);
 delay(1);
}
```

**Receiver Side:**

```
#include <ELECHOUSE_CC1101_2560.h>

 void setup()
{
 Serial.begin(9600);
 ELECHOUSE_cc1101.Init();
 ELECHOUSE_cc1101.SetReceive();
}

byte RX_buffer[1]={0};
byte size,i,flag;

void loop()
{
 if(ELECHOUSE_cc1101.CheckReceiveFlag())
 {
   size=ELECHOUSE_cc1101.ReceiveData(RX_buffer);
   for(i=0;i<size;i++)
   {
     Serial.print(RX_buffer[i],DEC);
     Serial.print(' ',BYTE);
```

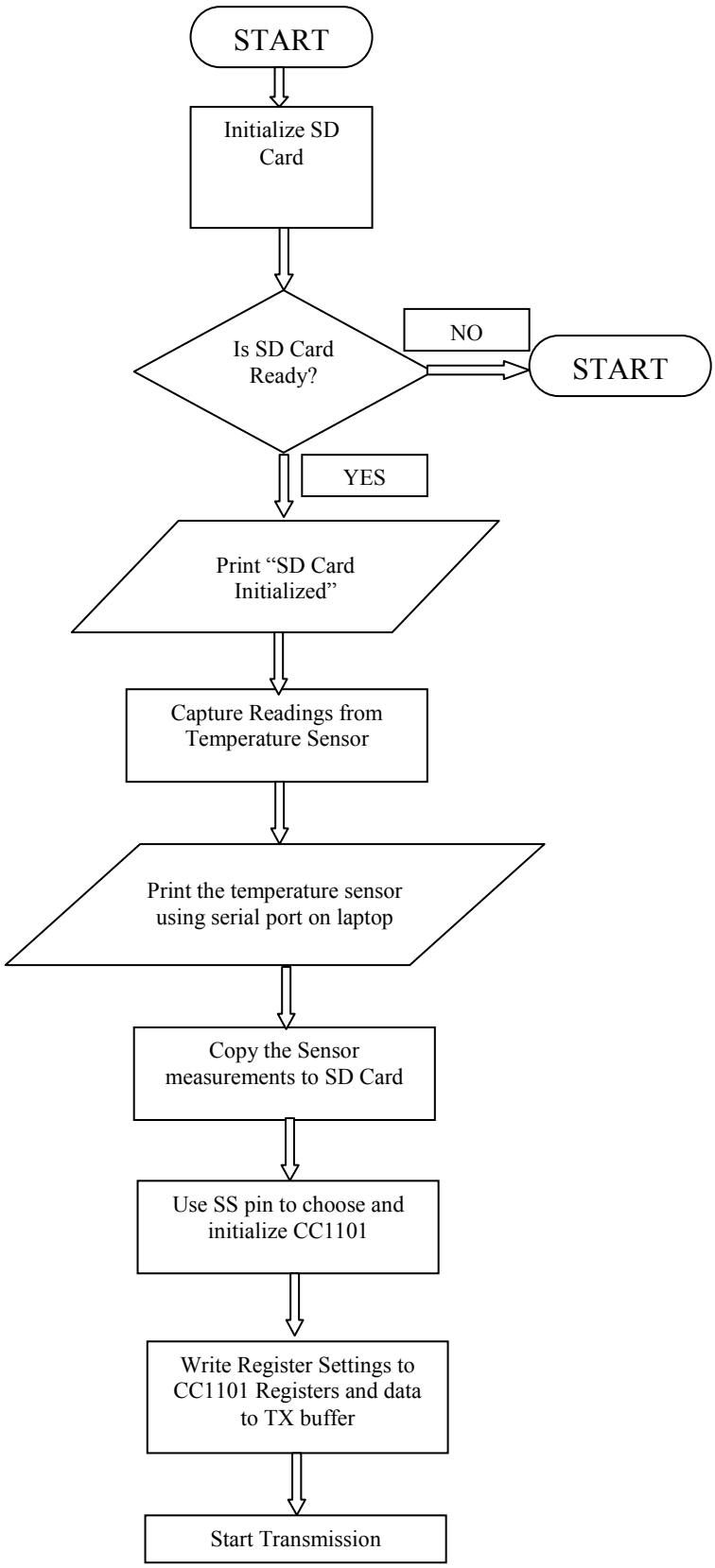
```
    }
    Serial.println("");
    ELECHOUSE_cc1101.SetReceive();
  }
}
```

**Parameters to be set on the CC1101 Registers Mandatorily:**

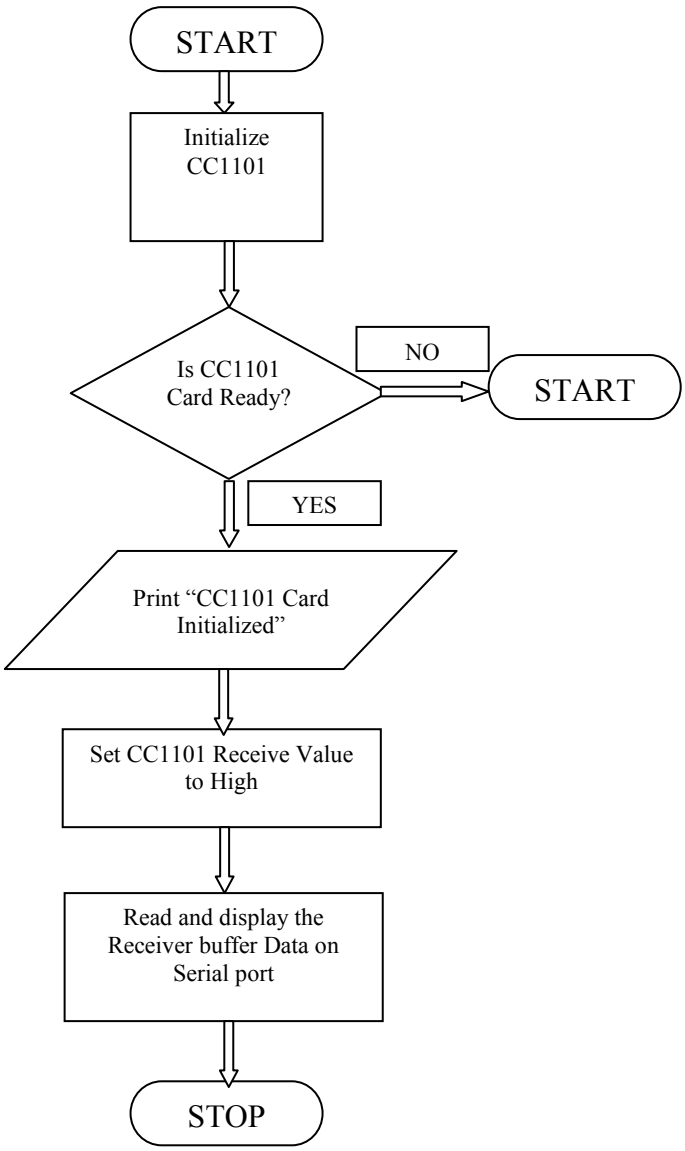
1).RF output power
2).RX filter bandwidth
3). Deviation
4). Datarate
5). Modulation
6). Manchester enable
7).RF Frequency
8). Channel spacing
9). Channel number / / Optimization
10_.Sync mode
11). Format of RX / TX data
12). CRC operation
13)./ Forward Error Correction
14). Length configuration
15)./ Packetlength
16) Preamble count
17). Append status
18). Address check
19).FIFO auto flush
20).Device address
21). GDO0 signal selection
22).GDO2 signal selection

# APPENDIX C – FLOW CHART

TRANSMITTER SIDE



# RECEIVER SIDE



# APPENDIX D – EQUIPMENT LIST

**1). Texas Instruments CC1101 Transceiver**

**2). Smart EF 04EB**

**4).Arduino Mega 2560**

**5). Arduino Uno**

**6).RS232 TTL Camera**

**7). Temperature Sensor DS1620**