DOCUMENTATION OF CODE FOR STEPWELL

#include <Stepper.h>

#include <Wire.h>

#include <SoftwareWire.h>

#include <RtcDS1307.h>

#include <SPI.h>

#include <SD.h>

#include <String.h>

These are all the necessary libraries included in the code to run different functions

#include <Stepper.h> is used for stepper motor control

#include <Wire.h> is used for I2C(inter-integrated circuit) communication. I2C is a two-wire protocol used to establish serial communication between two devices

#include <SoftwareWire.h> is used for software-based I2C communication protocol for Arduino

#include <RtcDS1307.h> is used for real-time clock

#include <SPI.h> is used for SPI(serial peripheral interface)communication. SPI is a synchronous serial communication that exchanges data between microcontrollers and peripherals devices like sensors, memory chips, etc.

#include <SD.h> is used for SD card data handling

#include <String.h> is used for string manipulation

DEFINING CONSTANTS AND VARIABLES

#define SDA 3: Represents the pin number used for the I2C data line (Serial Data).

#define SCL 2: Represents the pin number used for the I2C clock line (Serial Clock).

SoftwareWire myWire(SDA, SCL); to initialize the softwareWire with class myWire used for I2C interfacing on SDA and SCL pins

RtcDS1307<SoftwareWire> Rtc(myWire); is used to create an instance of RtcDS1307 class that uses software-based I2C interface for communication with DS1307 real-time clock module

#define gprsSerial Serial: it defines gprsSerial to use the built-in hardware serial for communicating with the GSM module.

float windup_ckt = 0; float variable is used to store the windup check value

const int stepsPerRevolution = 2048; number of steps for revolution for the stepper motor

#define IN1 7

#define IN2 6

#define IN3 5

#define IN4 4

Pins IN1,IN2,IN3,IN4 are defined for controlling the stepper motor

#define forcePin A7 : defines the force pin as A7 analog pin in Arduino

#define Switch A5: defines switch as A5 representing analog pin connected to switch for sensor input

#define chipSelect 10: defines chipselect as PIN 10 used for chip select when communicating with an SD card module

#define countof(a) (sizeof(a) / sizeof(a[0])) : defines countof(a) to calculate the number of elements in an array

float dist_per_step = 0.03068; it defines the stepper motor moves per step. This value is calculated based on the circumference of the spool and the number of steps per revolution.

int stepval = 162; it represents the number of steps to be executed each time when the motor moves

int reading = 0; firstly reading value is declared as 0 it stores the sensor readings

float distance = 0; floating point variable is initialized as 0

unsigned long int reset_time = millis(); it represents the number of milliseconds since the Arduino board began running

Stepper myStepper(stepsPerRevolution, IN1, IN3, IN2, IN4); initializing the stepper with object name my stepper with parameters specifying the number of steps per revolution and the pin numbers (IN1, IN2, IN3, IN4) connected to the stepper motor.

void writeData(float distance, const RtcDateTime& dt){ : Defines a function writeData that takes a float distance and a const RtcDateTime& dt as parameters.

File dataFile; declares the file with the name datafile

dataFile = SD.open("datalog.txt", FILE_WRITE); Opens or creates the file "datalog.txt" on the SD card for writing (FILE_WRITE mode) and assigns it to dataFile.

if (dataFile) { : checks if the data file opened or not

char datestring[26]; declares a character array “datestring” to store the date

snprintf_P(datestring,

countof(datestring),

PSTR("%04u-%02u-%02uT%02u:%02u:%02u+00:00"),

dt.Year(),

dt.Month(),

dt.Day(),

dt.Hour(),

dt.Minute(),

dt.Second());

these functions are used to format the date and time into the datestring using a specified format string and write the formatted date, time, and distance values into the dateFile

dataFile.print(datestring);

dataFile.print(" ");

dataFile.print("distance : ");

dataFile.println(distance);

using these print functions it will print the date, time and distance

dataFile.close(); closes the datafile to save changes into the SD card

delay(2000); giving 2000ms of delay time to ensure data is written in SD card

void send_data(float Distance_S) { this function will send data mainly Distance _S via GSM module using AT commands

gprsSerial.println("AT+CIPSHUT"); it will send an AT command (AT+CIPSHUT)to shutdown the IP stack

gprsSerial.println("AT+CIPSTATUS"); it will send an AT command (AT+CIPSTATUS) to check the status of the IP connection

gprsSerial.println("AT+CIPMUX=0"); it will send an (AT+CIPMUX=0) command to set single IP connection mode

ShowSerialData(); it will clear the serial input buffer to prepare the serial port for subsequent serial communication

gprsSerial.println("AT+CSTT=\"cmnet\""); it will send the AT command ("AT+CSTT=\"cmnet\""); to setup APN for the mobile network

ShowSerialData(); it will again clear the serial input buffer

gprsSerial.println("AT+CIICR"); it will send an AT command (AT+CIICR) to bring wireless connection

gprsSerial.println("AT+CIFSR"); it will send this AT command to get the local IP address

ShowSerialData(); it will again clear the serial input buffer

gprsSerial.println("AT+CIPSPRT=0"); it will send this AT command to set the connection type to transparent

ShowSerialData(); it will again clear the serial input buffer

gprsSerial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\""); it will send the AT command to start the TCP connection with the thingspeak server on port 80

ShowSerialData(); it will again clear the serial input buffer

gprsSerial.println("AT+CIPSEND"); it will this AT command to begin the process of sending data over TCP connection

ShowSerialData(); it will again clear the serial input buffer

gprsSerial.println("GET https://api.thingspeak.com/update?api_key=6FU7YFXWVWCNCECH&field1="+String(Distance_S));

it will send an HTTP GET request to the specified ThingSpeak URL to update a field(field1) with the value of Distance_S

ShowSerialData(); it will again clear the serial input buffer

gprsSerial.println((char)26); it will send the ASCII control Z character to singal the end of the HTTP request which was initiated with AT+ CIPSEND command.

gprsSerial.println(); it will send the newline character to terminate the current command and prepare for the next command

ShowSerialData(); it will again clear the serial input buffer

gprsSerial.println("AT+CIPSHUT"); it will send this AT command to close the TCP connection

ShowSerialData(); clears the serial input buffer to handle other responses after closing the TCP connection

delay(1000): after every gprsSerial function some amount of delay is given for executing the particular command.

void ShowSerialData(){ begins the definition of a function named showSerialData() that does not return any value

while(gprsSerial.available()!=0) Executes a loop as long as there is data available in the serial input buffer.

gprsSerial.read(); It reads and discards one byte of data from the serial input buffer

delay(5000); after clearing the serial buffer the program waits for 5secs

void printDateTime(const RtcDateTime& dt) { : Defines a function named printDateTime that takes a constant reference to a RtcDateTime object (dt) as its parameter and does not return a value

char datestring[20]; declares the character array with the name datestring with 20 characters to store formatted date and time.

snprintf_P(datestring,

countof(datestring),

snprintf_p is used to format date and time information into the string(datestring)

PSTR("%02u/%02u/%04u %02u:%02u:%02u"),

dt.Month(),

dt.Day(),

dt.Year(),

dt.Hour(),

dt.Minute(),

dt.Second());

according to the given format, %02u represents an unsigned integer format with 2 digits %04 represents an unsigned integer format with 4 digits

void rtc_setup(){ It defines a function named rtc_setup that initializes the real-time clock (RTC) module and sets its initial configuration.

Rtc.Begin(); It calls the Begin method on the Rtc object to initialize the RTC module.

RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__); it creates a new RtcDateTime object named compiled this sets an initial reference time for RTC

printDateTime(compiled); it is used to print the formatted date and time

if (!Rtc.IsDateTimeValid()) { It checks if the current date and time stored in the RTC module are valid are not.

Rtc.SetDateTime(compiled); If the RTC date and time are not valid it sets the RTC module's date and time to the compiled date and time

if (!Rtc.GetIsRunning()){ Checks if the RTC module is actively running or not.

Rtc.SetIsRunning(true); If the RTC module is not running it starts the RTC module

RtcDateTime now = Rtc.GetDateTime(); Retrieves the current date and time from the RTC module and stores it in the now variable.

Rtc.SetDateTime(compiled); If the current RTC time is earlier than the compiled time, sets the RTC time to the compiled time.

void setup() { initializing and setting up the sketch

myStepper.setSpeed(15); setting the stepper motor speed to 15 RPM

gprsSerial.begin(9600); Initializes the serial communication for the GSM module at a baud rate of 9600.

do {

windup_ckt = analogRead(Switch); Executes a loop that reads the analog value from the switch

myStepper.step(162); setting up the step value as 162

while (windup_ckt < 500.0);

distance = 0;

When the switch reading exceeds 500 the loop winds up and it goes to starting position and it resets the distance value to 0

void loop() { This function is called repeatedly after setup() function completes. It contains the main logic that executes in a loop in the Arduino

READING THE FORCE SENSOR:

reading = (reading + analogRead(forcePin))/2; Reads the analog value from the force sensor and calculates a moving average by adding the current reading to the previous value and dividing by 2.

```
while (reading > 10) {

myStepper.step(-162);

  delay(500);

distance += stepval * dist_per_step;
```

It enters a loop if the force sensor reading is greater than 10. Inside the loop, the stepper motor steps in reverse (-162 steps), and the distance measurement is updated based on the step value and distance per step

```
while (reading <=10) {

myStepper.step(162);

  delay(500);

  distance -= stepval * dist_per_step;
```

It enters a loop if the force sensor reading is less than or equal to 10. Inside the loop, the stepper motor steps forward (162 steps), and the distance measurement based on the step value and distance per step

send_data(distance); This function is used to transmit the distance value over GSM using AT commands

rtc_setup(); This function is used to initialize and configure the real-time clock (RTC) module

writeData(distance,RtcDateTime(__DATE__, __TIME__)); This function is used to write the current distance value and current date/time to an SD card file

delay(5 * 60000); giving 5 minutes of delay time

```
if (millis() - reset_time >= 3 * 60 * 60000) {

   do {

     windup_ckt = analogRead(Switch);

     myStepper.step(-162);

   } while (windup_ckt < 500.0);

   distance = 0;

  }

}
```

Checks if the elapsed time since the last reset is greater than or equal to 3 hours (3 * 60 * 60,000 milliseconds). If true, it enters a loop to wind up the stepper motor until the switch threshold is reached and then resets the distance measurement to zero.