REPORT

On

GPS interfacing with ATMEGA328p

For

ECE342:EMBEDDED SYSTEM ARCHITECTURE AND PROGRAMMING

Submitted by:

Harsh Bhadani	12014525	21
Siddharth Mehrotra	12006050	14
Riya Verma	12003905	07
Nandini Singh	12016334	23

Submitted to: Mohd Wasim

Declaration

We hereby declare that we have completed the project “GPS interfacing with ATMEGA328p” with the valuable guidance of Mohd Wasim sir. We have worked with utmost dedication for this project and the learning outcomes fulfill the requirements of the project for the award of the degree of B.Tech. in Electronics and Communication Engineering (ECE), Lovely Professional University, Phagwara.

Sincerely,

Team Members

Acknowledgment

I would like to express my sincere gratitude to all those who have contributed to the successful completion of the project “GPS interfacing with ATMEGA328p ”.

Firstly, I would like to thank my course supervisor Mohd Wasim sir. , for providing me with invaluable guidance and support throughout the project. Without their constant motivation and direction, this project would not have been possible.

I would also like to extend my appreciation to all members, who provided me with access to their resources and infrastructure. This enabled me to gather the necessary data and information required for the successful implementation of the project.

Furthermore, I am grateful to my classmates and colleagues for their support and encouragement. Their feedback and suggestions have been instrumental in improving the quality of this project.

Lastly, I would like to acknowledge the invaluable support of my family and friends, who have always been there for me, providing me with their unwavering love and support.

Thank you all for your contributions and support, without which this project would not have been possible.

Sincerely,

Team Members

Aim of the project

The aim of the project is to interface the GPS module with ATMEGA328p to get longitudinal as well as latitudinal coordinates and display them on the serial monitor of Arduino UNO.

Introduction

Electronic devices often use GPS modules to track the location using longitude and latitude coordinates. Applications of GPS include vehicle tracking systems, GPS clocks, accident detection alert systems, traffic navigation, surveillance systems, etc. GPS uses data from many satellites to provide altitude, latitude, longitude, UTC time, and other location-specific information. A microcontroller is required to read data from GPS and in this project, we are using ATMEGA328p.

GPS

The Global Positioning System (GPS) uses signals transmitted by satellites in orbit and ground stations on Earth to pinpoint a user's location on the planet.

The GPS receives radio frequency signals that are transmitted by satellites and ground stations. These signals are used by GPS to pinpoint their precise location.

GPS doesn't transmit any data. The timestamps of the time the signals were transmitted are included in the signals that are received from satellites and ground stations. The distance between the satellites and the GPS may be calculated using a straightforward formula for distance using speed and time by calculating the time difference between the time the signal was transmitted and the time the signal was received and using the speed of the signal.

The GPS can be used to triangulate its exact location using data from three or more satellites.

The GPS Module will broadcast data at a 9600 Baud rate in several strings. The GPS data can be seen using a UART port with a 9600 Baud rate.

In NMEA format, the GPS module transmits real-time tracking position data (see the aforementioned screenshot). There are multiple sentences in the NMEA format, including four crucial sentences.

- $GPGGA: Global Positioning System Fix Data
- $GPGSV: GPS satellites in view
- $GPGSA: GPS DOP and active satellites
- $GPRMC: Recommended minimum specific GPS/Transit data

Mostly $GPGGA string is used for tracking any location.

To understand these NMEA commands, another code for decoding these commands is used.

The GPS receiver module uses USART communication to communicate with the controller.

Components required

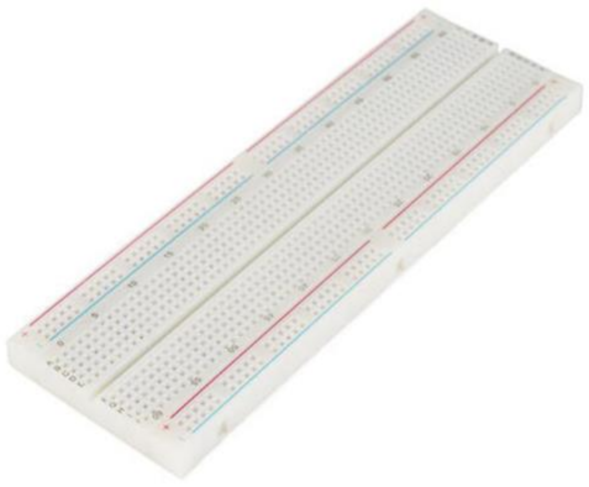
1. ATMEGA 328

The ATmega328 is a single-chip microcontroller created by Atmel in the mega AVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core.NEO-6M GPS Module

Ublox Neo 6M is a serial GPS module that provides location details through serial communication. This module has an external antenna, built-in EEPROM, and four pins:

Pin	Description
Vcc	2.7 – 5V power supply
Gnd	Ground
TXD	Transmit Data
RXD	Receive Data

3. Breadboard



A breadboard is a reusable device used for prototyping and testing electronic circuits. It is a rectangular plastic board with a series of holes arranged in a grid pattern, and each hole is connected to one or more metal strips that run beneath the board's surface. The strips are arranged in columns, with each column electrically connected to the others.

4. Jumper wires

Jumper wires are used to connect components on a breadboard or circuit board. In this project, Male to Female jumper wires are used to connect the components in the transmitter and receiver circuits.

Code

Code for receiving and displaying the longitude and latitude

```
#include <SoftwareSerial.h>

// The serial connection to the GPS module

SoftwareSerial ss(4, 3);  //pin 4 and pin 3 as RX and TX

void setup(){

  Serial.begin(9600);

  ss.begin(9600);

}


void loop(){

  while (ss.available() > 0){

    // get the byte data from the GPS

    byte gpsData = ss.read();

    Serial.write(gpsData);

  }

}
```

Code for decoding the NMEA signals

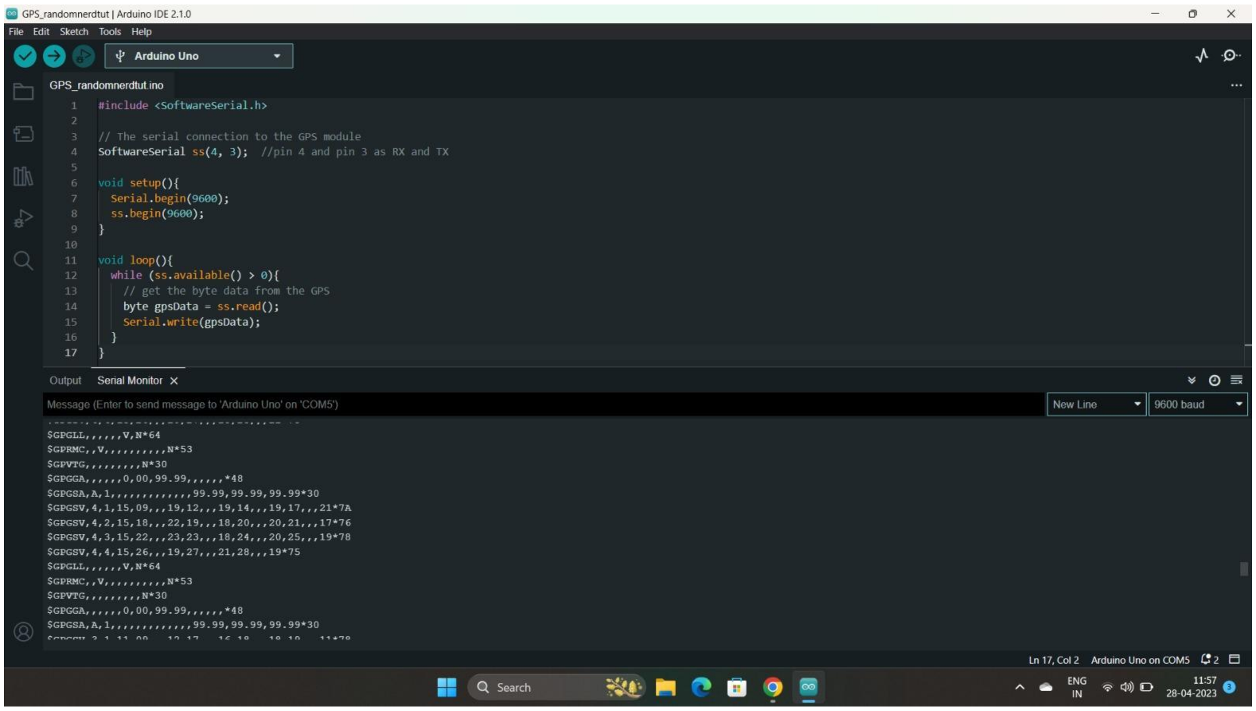
```
#include <TinyGPSPlus.h>

/*

   This sample sketch should be the first you try out when you are testing a TinyGPSPlus

   (TinyGPSPlus) installation.  In normal use, you feed TinyGPSPlus objects characters
from

   a serial NMEA GPS device, but this example uses static strings for simplicity.

*/



// A sample NMEA stream.

const char *gpsStream =

  "$GPRMC,045103.000,A,3014.1984,N,09749.2872,W,0.67,161.46,030913,,,A*7C\r\n"

  "$GPGGA,045104.000,3014.1985,N,09749.2873,W,1,09,1.2,211.6,M,-
22.5,M,,0000*62\r\n"

  "$GPRMC,045200.000,A,3014.3820,N,09748.9514,W,36.88,65.02,030913,,,A*77\r\n"

  "$GPGGA,045201.000,3014.3864,N,09748.9411,W,1,10,1.2,200.8,M,-
22.5,M,,0000*6C\r\n"

  "$GPRMC,045251.000,A,3014.4275,N,09749.0626,W,0.51,217.94,030913,,,A*7D\r\n"

  "$GPGGA,045252.000,3014.4273,N,09749.0628,W,1,09,1.3,206.9,M,-
22.5,M,,0000*6F\r\n";



// The TinyGPSPlus object
```

```
TinyGPSPlus gps;


void setup()

{

  Serial.begin(115200)

  Serial.println();


  while (*gpsStream)

    if (gps.encode(*gpsStream++))

      displayInfo();


  Serial.println();

  Serial.println(F("Done."));

}


void loop()

{

}


void displayInfo()
```

```
{
  Serial.print(F("Location: "));
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.print(F("  Date/Time: "));
  if (gps.date.isValid())
  {
    Serial.print(gps.date.month());
    Serial.print(F("/"));
    Serial.print(gps.date.day());
    Serial.print(F("/"));
  Serial.print(gps.date.year());
}
else
{
  Serial.print(F("INVALID"));
}

Serial.print(F(" "));
if (gps.time.isValid())
{
  if (gps.time.hour() < 10) Serial.print(F("0"));
  Serial.print(gps.time.hour());
  Serial.print(F(":"));
  if (gps.time.minute() < 10) Serial.print(F("0"));
  Serial.print(gps.time.minute());
  Serial.print(F(":"));
  if (gps.time.second() < 10) Serial.print(F("0"));
  Serial.print(gps.time.second());
  Serial.print(F("."));
  if (gps.time.centisecond() < 10) Serial.print(F("0"));
```

```
    Serial.print(gps.time.centisecond());
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.println();
}
```

Output



GPS_randomnerdtut | Arduino IDE 2.1.0

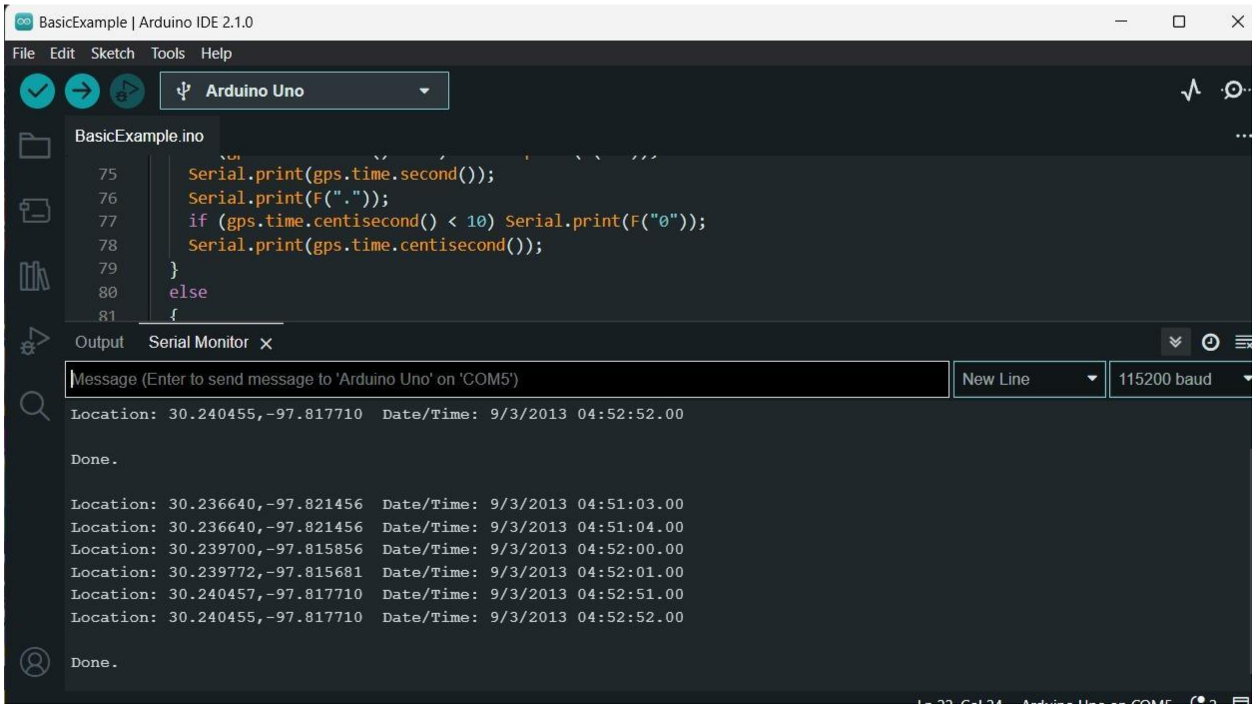
```
#include <SoftwareSerial.h>

// The serial connection to the GPS module
SoftwareSerial ss(4, 3);  //pin 4 and pin 3 as RX and TX

void setup(){
  Serial.begin(9600);
  ss.begin(9600);
}

void loop(){
  while (ss.available() > 0){
    // get the byte data from the GPS
    byte gpsData = ss.read();
    Serial.write(gpsData);
  }
}
```

Message (Enter to send message to 'Arduino Uno' on 'COM5') New Line 9600 baud

```
$GPGLL,,,,,,V,N*64
$GPRMC,,V,,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30
$GPGGA,,,,,,0,00,99.99,,,,,,*48
$GPGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99*30
$GPGSV,4,1,15,09,,,19,12,,,19,14,,,19,17,,,21*7A
$GPGSV,4,2,15,18,,,22,19,,,18,20,,,20,21,,,17*76
$GPGSV,4,3,15,22,,,23,23,,,18,24,,,20,25,,,19*78
$GPGSV,4,4,15,26,,,19,27,,,21,28,,,19*75
$GPGLL,,,,,,V,N*64
$GPRMC,,V,,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30
$GPGGA,,,,,,0,00,99.99,,,,,,*48
$GPGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99*30
```

Ln 17, Col 2 Arduino Uno on COM5

Output after decoding

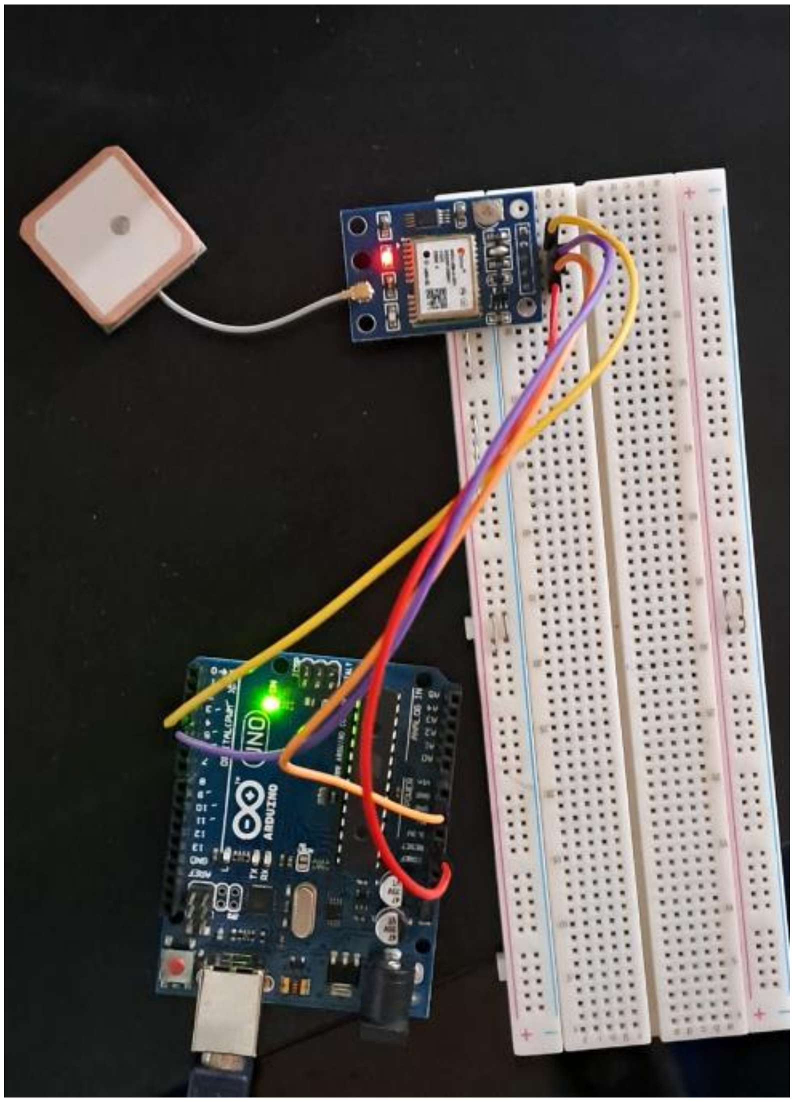


BasicExample | Arduino IDE 2.1.0

```
      Serial.print(gps.time.second());
      Serial.print(F("."));
      if (gps.time.centisecond() < 10) Serial.print(F("0"));
      Serial.print(gps.time.centisecond());
    }
    else
    {
```

Message (Enter to send message to 'Arduino Uno' on 'COM5') New Line 115200 baud

```
Location: 30.240455,-97.817710  Date/Time: 9/3/2013 04:52:52.00

Done.

Location: 30.236640,-97.821456  Date/Time: 9/3/2013 04:51:03.00
Location: 30.236640,-97.821456  Date/Time: 9/3/2013 04:51:04.00
Location: 30.239700,-97.815856  Date/Time: 9/3/2013 04:52:00.00
Location: 30.239772,-97.815681  Date/Time: 9/3/2013 04:52:01.00
Location: 30.240457,-97.817710  Date/Time: 9/3/2013 04:52:51.00
Location: 30.240455,-97.817710  Date/Time: 9/3/2013 04:52:52.00

Done.
```



Conclusion

The GPS module can be used to transmit longitudinal and latitudinal signals and in this project, the signals were displayed on Arduino UNO using UART communication.