Development of GPS Assisted Online CO2-Temperature Mapping System

GPS を利用した CO2 と気温のマッピングシステムの開発

March 2014

Thammita A.S. Anuruddha

Graduate School Of Science and Engineering
Ibaraki University,
Hitachi City

Submitted As a Partial Requirement of PhD in Applied Beam Science

Abstract

The objective of this research was to develop a global access method for accurate CO2 density in certain points of the world. The greenhouse effect is one of most potential deleterious impact of human race and all other species on earth. This is a very big concern on the scientific field as well as it seems to be a good reason even for changing governments, being a strong political issue today. Emerging of greenhouse gases as a result of uncontrolled development makes a world an unsuitable place to live for existences. Gaseous constituents that absorb and emit radiation at specific wavelengths within the spectrum of thermal infrared radiation are known as Greenhouse Gases (GHGs) (Scheutz et al., 2009). Most of world agreed and accepts that a human-induced increasing concentration of GHG in the atmosphere of the earth causes global climate change. Direct and indirect impacts of industrialization such as fossil fuel burning etc.., enhanced the emission of green house gases such as CO2 after the 20th Century. As of January 2011, the level of atmospheric CO2 is monitored as 391.19 ppm by volume. It is very helpful to measure CO2 level and temperature level with GPS information because it can easily map the both levels of a particular area.

Properly designed and managed online knowledge sharing systems can improve the availability of real-time data around the world through internet among interested users of such data as researchers, students, decision makers etc. The data gathered by the sensing device stored in such knowledge sharing system called KISSEL (Knowledge Integrated Servers System for E-Learning) which operated and maintained by Ibaraki University, Japan. Accessing to KISSEL, anyone can monitor certain existence CO2 levels and densities in the places where sensing devices are installed. A prototype has been developed for a versatile, flexible, cost efficient, and high speed instrument to monitoring the CO2 over Temperature. It helps to map CO2 data of a path such as roads, sea paths, and air ways below 18 km altitude or a place for long time. The device is named as –AIR BOY–. This system is portable and easy to use. The data collected and stored in this system alone can be used to analyze CO2 concentrations against time, temperature, global coordinates or altitude. It utilizes a CO2 Sensor, temperature Sensor, a GPS+Altitude receiver, LCD Panel, a microcontroller and a USB-UART module with an Ethernet interface. Gathered data by the system, transmits to a database in a server called KISSEL.

Gathered data transmitted from the sensing device to the database over internet by http GET request, with encapsulated data of its quarry string. In the end of the server side, it de-capsulate the quarry string in to data and stores it in the database. Then it mapped on a Google map according to a color code in respect of the density of CO2 in each point. This system can be deployed anywhere in the world whether internet connection is available or not. Once it connects to the internet, it starts to send the saved data to the KISSEL data base and update it.

Power consumption is also very low of this module can simply operate using 1x1 feet solar panel. Internal rechargeable battery pack can store power to operate overnight. Taking In-Situ data of GHG can be very useful in industrial areas, and garbage dumps etc which cannot be taken using remote sensing method using satellite spectrum images. It is very useful for respective parties such as researchers and general public. In this study, it is gathered data in three countries, Sri Lanka, India and Japan, observed a higher level of CO2 around industrial, urban areas than rural areas in all three countries. Temperature also was higher in such places than rural areas. Multiple place monitoring in same time of the day in each region is necessary. This system can fix as many as possible in some industrial areas to gather data and share that data to the general public without any fee. Exposing this unknown data of GHG emissions free to public can give them an idea to force their respective community leaders, company owners, vehicle users to reduce the emission level of GHG to reduce the rise of climate change. This study demonstrates that system could make meaningful contributions to global climate-change mitigation by making its real-time sensed data available to public and other respective parties. It should be further developed by adding a good and solid body; a water resistant option should also be added.

Acknowledgement

A major research project like this is never the work of anyone alone. The contributions of many different people, in their different ways, have made this possible. I would like to extend my appreciation especially to the following.

I sincerely Thank Prof, Atsushi Minato and Satoru Ozawa for making this research possible. their support, guidance, advices throughout the research project, as well as his pain-staking effort in proof reading the drafts, are greatly appreciated. Indeed, without their guidance, I would not be able to put the topic together.

My good Friend Ravi, for encouraging me to undertake the master's and PhD programs. The experience has been an interesting and rewarding one. Thanks Ravi.

Of course, this project would not have been possible without the participation of the Bhaggya, who helped me lot during this project by developing many modules of the system.

All the Srilankan fellowmen in Japan, who shared happiness and burdens for long time. Thank you pals.

My colloquies in the Lab, Warnajith, Youe San, Gihan And Lakesh, who did a great help when I was not in Japan during the research. Thanks mates.

Ibaraki University, Japanese Government and people of Japan, Thank you giving this opportunity to me.

Last but not least, I would like to thank my family members for their unconditional support, both financially and emotionally throughout all of my degrees. In particular, the patience and understanding shown them during the fourteen years up to complete my PhD, is greatly appreciated. I know, at times, my temper is particularly trying.

Preface

This thesis is submitted in partial fulfillment of the requirements for a Doctor of Engineering in Applied Beam Science. It contains work done from March 2009. My supervisor on the project has been Professor Atsushi Minato, the Kansei Mathematic Laboratory in Ibaraki University. The dissertation has been made solely by the author; some of the text and figures however, is based on the research of others, and I have done my best to provide respective references to related sources.

In winter, 2008, my Professor Atsushi Minato introduced me the Microcontroller, with one of his simple development. That device intended for Reading GPS coordination, and processing data which operated solely by stand alone without a computer. I was impressed and intrigued by this new means of controlling without a computer, so I decided to develop some system using this device and I started to Develop "AIRBOY" for My masters Studies, later developed it as a full and completed system and a partial submission for my PhD.

Writing this thesis has been hard but in the process of writing I feel I have learned a lot and our conceptions of Environment Monitoring in Sri Lanka have certainly changed! I have dealt with a lot of subjects, in an attempt to give this thesis a broad perspective on Carbon Dioxide Monitoring in South Asia, thus combining many aspects of Applied Beam Science, Digital Electronics and human-computer interaction

Index Of Topics

Index of Figures

Bibiliography

1.0 AIRBOY and GPS

Smog hanging over cities is the most familiar and obvious form of air pollution. Mainly CO_2 is main greenhouse gas which liable for warming of the earth and then the water vapor. Industrialization enhanced the emission of green house gases such as CO_2.

It is very helpful to measure CO_2 level and temperature level with GPS information because it can easily map the both levels of a particular area.

This thesis is for introduce a prototype of developed architecture for a versatile, flexible, cost efficient, and high speed Instrument to monitoring the **CO_2** and temperature which helps to map data of a path such as roads, Sea paths, Air ways below 18 km altitude etc. The device is named as –AIR BOY– . In first and second chapters, it has been briefly introduced and explained, what is GPS, **CO_2**, and its related factors such as temperature, global warming etc.

1.1 What is GPS?

The Global Positioning System (GPS) is a satellite-based navigation system made up of a network of 24 satellites placed into orbit by the U.S. Department of Defense.

‘The Global Positioning System (GPS) is a U.S.-owned utility that provides users with positioning, navigation, and timing (PNT) services. This system consists of three segments: the space segment, the control segment, and the user segment. The U.S. Air Force develops, maintains, and operates the space and control segments.’ (National Coordination Office for Space-Based Positioning, Navigation, and Timing (2013-sep) What is GPS? [Online], Available: http://www. http://www.gps.gov/systems/gps/ 5,Nov,2013)

It was established in 1973 to overcome the limitations of previous navigation systems. GPS was originally intended for military applications, but in the 1980s, the government made the system available for civilian use. GPS works in any weather conditions, anywhere in the world, 24 hours a day. There are no subscription fees or setup charges to use GPS. It is a space-based global navigation satellite system (GNSS) that provides reliable location and time information in all weather and at all times and anywhere on or near the Earth when and where there is an unobstructed line of sight to four or more GPS satellites.

1.2 History of GPS

The design of GPS is based partly on similar ground-based radio navigation systems, such as LORAN and the Decca Navigator developed in the early 1940s, and used during World War II. In 1956 Friedwardt Winterberg proposed a test of general relativity using accurate atomic clocks placed in orbit in artificial satellites. To achieve accuracy requirements, GPS uses principles of general relativity to correct the satellites' atomic clocks. Additional inspiration for GPS came when the Soviet Union launched the first man-made satellite, Sputnik in 1957. A team of U.S. scientists led by Dr. Richard B. Kershner were monitoring Sputnik's radio transmissions. They discovered that, because of the Doppler Effect, the

frequency of the signal being transmitted by Sputnik was higher as the satellite approached, and lower as it continued away from them. They realized that because they knew their exact location on the globe, they could pinpoint where the satellite was along its orbit by measuring the Doppler distortion (see Transit (satellite)).

The first satellite navigation system, Transit, used by the United States Navy, was first successfully tested in 1960. It used a constellation of five satellites and could provide a navigational fix approximately once per hour. In 1967, the U.S. Navy developed the Timation satellite that proved the ability to place accurate clocks in space, a technology required by GPS. In the 1970s, the ground-based Omega Navigation System, based on phase comparison of signal transmission from pairs of stations,[3] became the first worldwide radio navigation system. Limitations of these systems drove the need for a more universal navigation solution with greater accuracy.

1.3 Why do we need GPS

While originally a military project, GPS is considered a dual-use technology, meaning it has significant military and civilian applications.
GPS has become a widely deployed and useful tool for commerce, scientific uses, tracking, and surveillance. GPS's accurate time facilitates everyday activities such as banking, mobile phone operations, and even the control of power grids by allowing well synchronized hand-off switching. Farmers, surveyors, geologists, and countless others perform their work more efficiently, safely, economically, and accurately.

Many civilian applications use one or more of GPS's three basic components: absolute location, relative movement, and time transfer.

1.4 How GPS works

“The Way it works is simple. Heavenly Bodies appear to travel in predictable path across the sky. With Certain tools – Sexent, astrolabe Chronometer experienced navigators could measure relative angles and distances between themselves and these reference points giving them a fairly good idea of where they were on earth.” (Steve Featherstone - Outdoor Guide to using your GPS – Page 09, 2004)

GPS satellites circle the earth twice a day in a very precise orbit and transmit signal information to earth. GPS receivers take this information and use triangulation to calculate the user's exact location. Essentially, the GPS receiver compares the time a signal was transmitted by a satellite with the time it was received. The time difference tells the GPS receiver how far away the satellite is. Now, with distance measurements from a few more satellites, the receiver can determine the user's position and display it on the unit's electronic map.

A GPS receiver must be locked on to the signal of at least three satellites to calculate a 2D position (latitude and longitude) and track movement. With four or more satellites in view,

the receiver can determine the user's 3D position (latitude, longitude and altitude). Once the user's position has been determined, the GPS unit can calculate other information, such as speed, bearing, track, trip distance, distance to destination, sunrise and sunset time and more.

1.5 How accurate GPS is?

Today's GPS receivers are extremely accurate, thanks to their parallel multi-channel design. Garmin's 12 parallel channel receivers are quick to lock onto satellites when first turned on and they maintain strong locks, even in dense foliage or urban settings with tall buildings. Certain atmospheric factors and other sources of error can affect the accuracy of GPS receivers. AIRBOY's GPS receiver is accurate to within 15 meters on average.

1.6 The GPS satellite system

The 24 satellites that make up the GPS space segment are orbiting the earth about 12,000 miles above us. They are constantly moving, making two complete orbits in less than 24 hours. These satellites are traveling at speeds of roughly 7,000 miles an hour.

GPS satellites are powered by solar energy. They have backup batteries onboard to keep them running in the event of a solar eclipse, when there's no solar power. Small rocket boosters on each satellite keep them flying in the correct path.

Here are some other interesting facts about the GPS satellites (also called NAVSTAR, the official U.S. Department of Defense name for GPS):

- The first GPS satellite was launched in 1978.
- A full constellation of 24 satellites was achieved in 1994.
- Each satellite is built to last about 10 years. Replacements are constantly being built and launched into orbit.
- A GPS satellite weighs approximately 2,000 pounds and is about 17 feet across with the solar panels extended.
- Transmitter power is only 50 watts or less.

1.7 What's GPS signals are?

Each GPS satellite transmits data on two frequencies, L1 (1575.42 Mhz UHF band) and L2 (1227.60 MHz). The signals travel by line of sight, meaning they will pass through clouds, glass and plastic but will not go through most solid objects such as buildings and mountains.

1.8 Sources of GPS signal errors

Factors that can degrade the GPS signal and thus affect accuracy include the following:

1.8.1 Ionosphere and troposphere delays - The satellite signal slows as it passes through the atmosphere. The GPS system uses a built-in model that calculates an average amount of delay to partially correct for this type of error.

1.8.2 Signal multipath - This occurs when the GPS signal is reflected off objects such as tall buildings or large rock surfaces before it reaches the receiver. This increases the travel time of the signal, thereby causing errors.

1.8.3 Receiver clock errors - A receiver's built-in clock is not as accurate as the atomic clocks onboard the GPS satellites. Therefore, it may have very slight timing errors.

1.8.4 Orbital errors - Also known as ephemeris errors, these are inaccuracies of the satellite's reported location.

1.8.5 Number of satellites visible - The more satellites a GPS receiver can "see," the better the accuracy. Buildings, terrain, electronic interference, or sometimes even dense foliage can block signal reception, causing position errors or possibly no position reading at all. GPS units typically will not work indoors, underwater or underground.

1.8.6 Satellite geometry/shading - This refers to the relative position of the satellites at any given time. Ideal satellite geometry exists when the satellites are located at wide angles relative to each other. Poor geometry results when the satellites are located in a line or in a tight grouping.

1.8.7 Intentional degradation of the satellite signal - Selective Availability (SA) is an intentional degradation of the signal once imposed by the U.S. Department of Defense. SA was intended to prevent military adversaries from using the highly accurate GPS signals. The government turned off SA in May 2000, which significantly improved the accuracy of civilian GPS receivers.

1.9 Other competing earth mapping systems

- The Russian GLObal NAvigation Satellite System (**GLONASS**)
- Chinese Compass navigation system (Planned)
- Galileo positioning system of the European Union (Planned).

1.10 Restrictions on civilian use

The U.S. Government controls the export of some civilian receivers. All GPS receivers capable of functioning above 18 kilometers (11 mi) altitude and 515 metres per second (1,001 kn)[48] are classified as munitions (weapons) for which U.S. State Department export licenses are required. These limits attempt to prevent use of a receiver in a ballistic missile. They would not prevent use in a cruise missile because their altitudes and speeds are similar to those of ordinary aircraft. AIRBOY GPS module is a civilian receiver.

1.11 Introduction to CO_2 Carbon Dioxide

1.12 What is Carbon dioxide ?

Carbon dioxide (chemical formula CO_2) is a chemical compound composed of two oxygen atoms covalently bonded to a single carbon atom. It is a gas at standard temperature and pressure and exists in Earth's atmosphere in this state. CO_2 is a trace gas comprising 0.039% of the atmosphere.
Deforestation and forest degradation accounts for 6-17% of allanthropogenic greenhouse gas emissions and tropical forest covercontinues to decline globally although at a slower rate than in thepast (FAO, 2010a;Hett, Castella, Heinimann, Messerli, & Pfund,2012;Le Quere, Raupach, Canadell, & Marland, 2009;Van derWerf et al., 2009).

As part of the carbon cycle known as photosynthesis, plants, algae, and cyanobacteria absorb carbon dioxide, sunlight, and water to produce carbohydrate energy for themselves and oxygen as a waste product. By contrast, during respiration they emit carbon dioxide, as do all other living things that depend either directly or indirectly on plants for food. Carbon dioxide is also generated as a by-product of combustion; emitted from volcanoes, hot springs, and geysers; and freed from carbonate rocks by dissolution.

As of January 2011, the level of atmospheric CO_2 is monitored as 391.19 ppm by volume. In 2009 January, it was 386.92, which monitored 4.27ppm has been grown within two years of period. Atmospheric concentrations of carbon dioxide fluctuate slightly with the change of the seasons, driven primarily by seasonal plant growth in the Northern Hemisphere. Concentrations of carbon dioxide fall during the northern spring and summer as plants consume the gas, and rise during the northern autumn and winter as plants go dormant, die and decay.

Carbon dioxide has no liquid state at pressures below 5.1 standard atmospheres (520 kPa). At 1 atmosphere (near mean sea level pressure), the gas deposits directly to a solid at temperatures below −78 °C (−108 °F; 195.1 K) and the solid sublimes directly to a gas above −78 °C. In its solid state, carbon dioxide is commonly called dry ice.

CO_2 is an acidic oxide: an aqueous solution turns litmus from blue to pink. It is the anhydride of carbonic acid, an acid which is unstable in aqueous solution, from which it cannot be concentrated. In organisms carbonic acid production is catalyzed by the enzyme, carbonic anhydrate.

1.13 History of CO_2

In the seventeenth century, the Flemish chemist Jan Baptist van Helmont observed that when he burned charcoal in a closed vessel, His interpretation was that the rest of the charcoal had been transmuted into an invisible substance he termed a "gas" or "wild spirit" (spiritus sylvestre).

In the 1750s by the Scottish physician Joseph Black found, limestone (calcium carbonate) could be heated or treated with acids to yield a gas he called "fixed air." He observed that the fixed air was denser than air and supported neither flame nor animal life.

Carbon dioxide was first liquefied (at elevated pressures) in 1823 by Humphry Davy

1.14 Greenhouse Effect

Carbon dioxide is a greenhouse gas as it transmits visible light but absorbs strongly in the infrared and near-infrared. As the chemically most stable non-condensing greenhouse gas, it acts as a critical 'climate control knob'.

1.15 CO_2 as a Greenhouse Gas

Since the beginning of the Industrial revolution, the burning of fossil fuels has increased the levels of carbon dioxide in the atmosphere from 280ppm to 391.19ppm. Unlike other pollutants, carbon dioxide emissions do not result from inefficient combustion: CO2 is a product of ideal, stoichiometric combustion of carbon. The emissions of carbon are directly proportional to energy consumption. CO2 is the second highest greenhouse gas in atmosphere.

Gas	Formula	Contribution (%)
Water Vapor	H_2O	36 – 72 %
Carbon Dioxide	CO_2	9 – 26 %
Methane	CH_4	4 – 9 %
Ozone	O_3	3 – 7 %

1.16 Global warming and Earth’s temperature

Local and global weather has always fluctuated and always will, so global warming cannot be expected to be a smooth process. But what can be seen above is that half of all man-made CO_2 has been put into the air since 1975, and that matches the one-degree F global temperature increase since 1975 rather well.

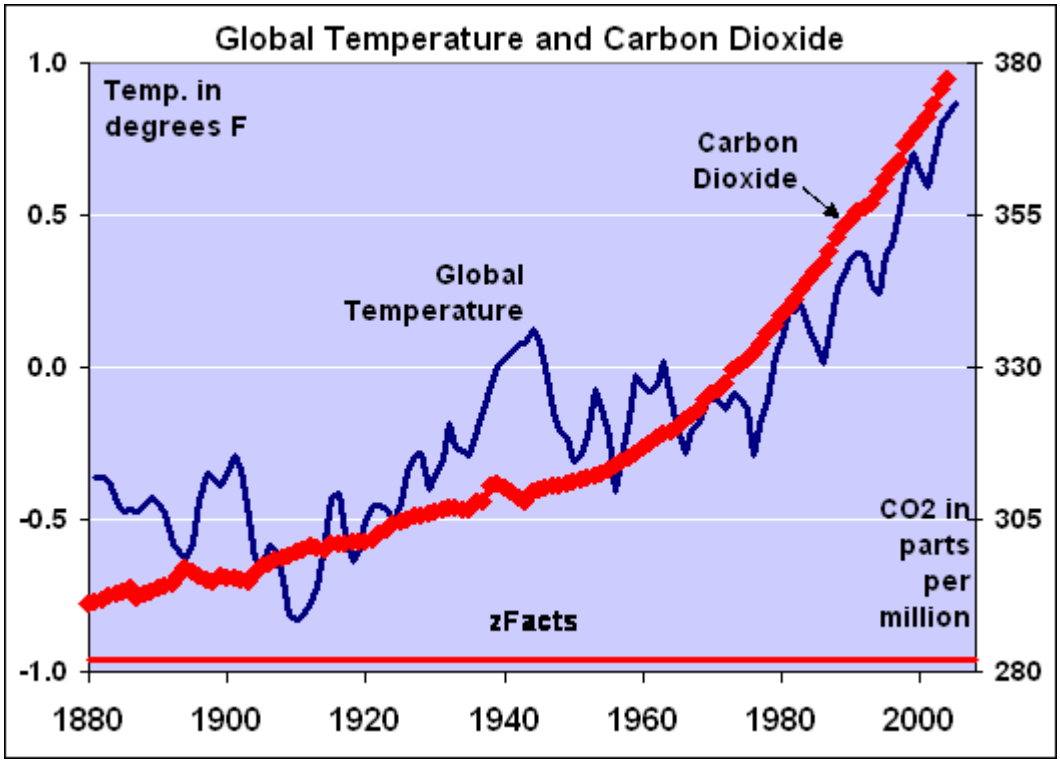


Figure 1 Global Temperature Over the Carbon Dioxide

Steven Stroft, Evidence that CO2 is Cause,[Online],
Available: http://www. http://zfacts.com/p/226.html [24 Aug 2013

2. CO_2 and AIRBOY

AirBoy is basically made for portable uses for any kind of user. It has several advantages over current existing systems. It Gathers CO_2 data with Temperature and integrates it with in-situ GPS coordination and store those data to EEPROM while storing it to the EEPROM. The system has several advantages.

2.1 The AIRBOY GPS based CO_2 and Temperature mapping system

The main objective was to develop air boy was to make a LOW-COST stand alone system for various kind of users such as researchers, travelers, etc.

The device is named as **–AIR BOY–** . This is a portable and easy to use device, because of its mobility and portability. Most of systems require PC and not standalone. Standalone systems are very costly comparing with **–AIR BOY–**. Its found that it is very hard to find a GPS based device which measures CO_2 and Temperature in the market.

2.2 How AIRBOY been composed.

This device been composed by using certain modules and protocols,

Devices

- **2.2.1 Arduino microcontroller**
- **2.2.2 CO_2 SenseAir K-30 temperature Sensor,**
- **2.2.3 GPS52D GPS receiver**
- **2.2.4 LCD Panel KL SN102 94V-0**
- **2.2.5 internal memory (EEPROM) ATMEL 24C1024**
- **2.2.6 Arduino Ethernet Shield DEV-09026**
- **2.2.7 SIMCom 908 GSM/GPRS Module**

Protocols

- **2.2.8 I^2C Bus**
- **2.2.9 UART**
- **2.2.10 TCPIP**
- **2.2.11 MODBUS**

- **What is Arduino?**

Arduino is an open-source computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can utilize to develop various modules, obtaining inputs from a variety of switches or sensors, and controlling a variety of circuitries such as , motors, and other outputs. Arduino projects can make as stand-alone, or they can be communicate with software running on the computer By downloading the open-source IDE (Bootloader), from internet, microcontroller can be make ARDUINO READY.

Arduino hardware consists an open source circuitry hardware based on 8 bit ATMEL AVR and 32 bit Atmel ARM core. In this project, it has been utilized the Atmel AVR architecture.

- **Why Arduino?**

Many other microcontrollers can found in the market and, many programmable platforms available for physical computing. Netmedia's BX-24, Parallax Basic Stamp, , MIT's Handyboard, Phidgets etc

According to ther Arduino Web page, "Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems" (http://arduino.cc/en/Guide/Introduction – Accessed 2012 Jan 15)

Arduino over the other Microcontrollers

Arduino has several advantages over other microcontrollers.

- Inexpensive
- Cross-platform
- Simple, clear programming environment
- Open source and extensible software
- Open source and extensible hardware

Arduino utilizes C++ programming architecture and very similer to c++ consist its own libraries to communicate with hardware. In robotics application, arduino can easily be used.

Arduino can be added various interchangeable modules as add-ons called shields. These shields can be connect to the hardware board through various pins or can be connected and addressed separately through I^2C serial Bus interface, a communication protocol developed by Philips Inc.

Original arduino utilizes Atmel AT MEGAfamily microcontrollers while a handful of some other brands been used by arduino compatibles. Usually the power input is 5V – 9V, included a 5V DC onboard power regulator and a 16 MHz crystal oscillator or a resonator made by ceramic.

Arduino microcontroller has a preprogrammed bootloader which make simplified the uploading process of the programme to its own on-chip flash memory.

All of arduino hardware can access through RS232 serial connection, however,

most of boards utilize a voltage level shifter in between computer and the hardware to change CMOS level to TTL levels vice versa. Without converting voltage levels properly, both computer and microcontroller cannot communicate. The level difference between TTL and CMOS is showed in following figure 3.1

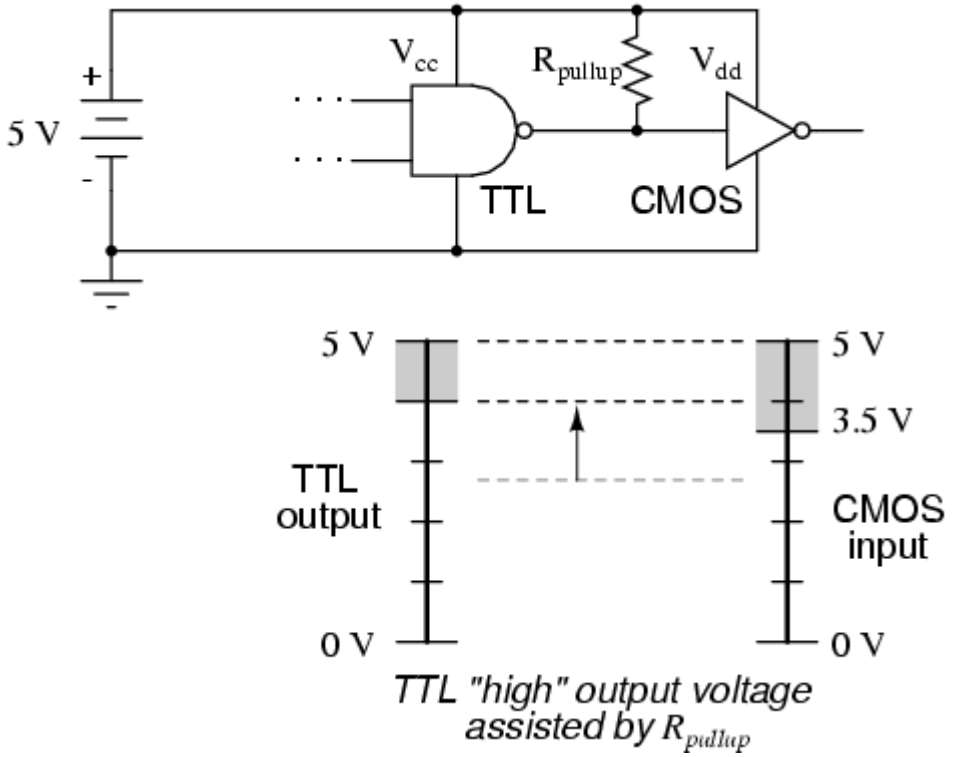


Figure 2 TTL and CMOS Logic Voltages

2.2.1 Arduino Microcontroller

The project is an effort to use the stand alone microcontroller of Arduino Uno (Atmega 328) with the other electronics parts in order to be portable and independent device. Atmega328 is an Atmel fabricated megaAVR series single-chip microcontroller, a high performance 8bit AVR RISC based one which has 32 KB ISP flash memory with programmable reading and writing ability. It consists following capabilities ad resources.

- 1 KB EEPROM
- 32 KB ISP flash memory
- 131 Powerful Instructions – Most Single Clock Cycle Execution
- 2 KB SRAM,
- 23 general purpose I/O lines

- Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
- Data retention: 20 years at 85°C/100 years at 25°C(1)
- 32 general purpose working registers
- Three flexible timer/counters with compare modes
- Internal and external interrupts,serial programmable USART
- A byte-oriented 2-wire serial interface
- SPI serial port
- 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages)
- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Programmable watchdog timer with internal oscillator
- Five software selectable power saving modes

The pin configuration of the microcontroller is as follows in figure 3

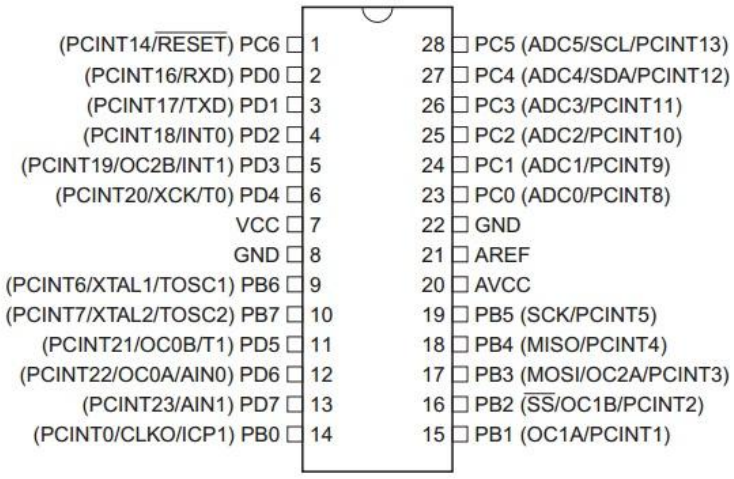


Figure 3 - pin configuration of the ATMega 328 microcontroller

Source – ATMEGA 328 Data Sheet, Atmel Corporation 2013,Aug

There are 28 pins available in this microcontroller including 23 general purpose I/O lines.

Pin Specification

Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

The Port B is an 8-bit I/O port with bi-directional capability and attached internal pull-up resistors (can be select for each bit alone). The output Port B buffers offers symmetrical drive characteristics with both high sink and source capability. Port B pins that are externally pulled low itself, will source current if the pull-up resistors are activated. The Port B pins are always tri-stated. This keeps it state when a reset condition occurred, even if the clock is not running.
By Depending on the fuse settings of clock selection, PB7 can be used as output from the inverting Oscillator amplifier. In the system, all of Port B pins are used for sensors and shields.

Port C (PC5:0)

Port C consists 7-bit bi-directional I/O ports with internal pull-up resistors (can be select for each bit alone). Symmetrical drive characteristics with both high sink and source capability are available in The PC5:0 output buffers. Port C pins that are pulled low will source current externally As inputs, if the pull-up resistors are activated. The Port C pins are always tri-stated. This keeps it state when a reset condition occurred, even if the clock is not running In the AIRBOY, all the Port C pins are utilized.

PC6/RESET

If the RSTDISBL Fuse is programmed, , PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the un-programmed RSTDISBL Fuse is available, PC6 is used as a Reset input. Keeping a low level state on this pin for longer than the minimum pulse length will generate a Reset, Shorter pulse length will not trigger the function however. In the system of AIRBOY, the reset pin has been utilized to reset the whole program. However, the external memory will not erase on this reset. All the flash registers erased by pressing the reset button of the system including GPS memory. After resetting, the system might take some time to receive GPS data and to determine the latitudes and longitudes which depends on the strength of signal available on site. Until proper GPS signal detected, the rest of system will not work such as temperature sensing, CO_2 sensing and Ethernet connection etc accords to the program hierarchy.

Port D

Port D consists 8-bit bi-directional I/O ports with internal pull-up resistors (can be select for each bit alone). Symmetrical drive characteristics with both high sink and source capability are available in The Port D output buffers. Port D pins that are pulled low will source current externally as inputs, if the pull-up resistors are activated. The Port D pins are always tri-stated. This keeps it state when a reset condition occurred, even if the clock is not running. In the system, all of PortD s are been deployed for sensors and shields.

AVCC

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. Even if the ADC is not used, It should be externally connected to VCC. If the ADC is used, it has been connected to VCC through a low-pass filter in the system. PC6..4 use digital supply voltage, VCC.

AREF

AREF is the analog reference pin for the A/D Converter. System utilizes this pin to sense analog data out from LM35 temperature sensor.

2.2.2 CO_2 SenseAir K-30 temperature Sensor

Figure 4 - SenseAir K-30 temperature Sensor

Air boy utilizes SENSEAIR K30 module as its CO_2 sensor, identified as a precise carbon dioxide detector for the project. The K30 sensor platform CO_2Engine® K30 can be customized for a variety of sensing and control applications such as AIRBOY. This platform is designed to be an OEM module for built-in applications in a host apparatus, and hence should be optimized for its tasks during a dialog between SenseAir and Arduino microcontroller. In this project, the task of receiving data has beed done successfully and Sense-Air kit works on arduino without any problem.

Functional Group	Descriptions and Ratings
Power Supply	
G+ referred to G0:	Absolute maximum ratings 4.5 to 14V, stabilized to within 10% 4.5 to 9V preferred operating range. Unprotected against reverse connection!
Serial Communication	
UART (TxD, RxD)	CMOS, ModBus communication protocol. Logical levels corresponds 3.3V powered logics. Refer to *"ModBus on* CO_2 *Engine K30"* for electrical specification.
Outputs	
OUT1	Buffered linear output 0..4 or 1..4VDC or 0..10V or 2..10V, depending on specified power supply and sensor configuration. ROUT < 100 _, RLOAD > 5 k_ **Load to ground only!** Resolution 10mV (8.5 bits in the range 0..4V)

OUT2	Buffered linear output 0..4 or 1..4VDC or 0..5V or 1..5V, depending on specified power supply and sensor configuration. ROUT < 100 _, RLOAD > 5 k_ **Load to ground only!** Resolution 5mV Can be used as alternative for OUT1, or for a second data channel, or in an independent linear control loop, such as a housing temperature stabilization
OUT3	CMOS **unprotected**. Digital (High/Low) output. High Output level in the range 2.3V min to DVDD = 3.3V. (1 mA source) Low output level 0.75V max (4 mA sink) Can be used for gas alarm indication, or for status indication etc.
OUT4	CMOS unprotected. Digital (High/Low) output. High Output level in the range 2.3V min to DVDD = 3.3V. (1 mA source) Low output level 0.75V max (4 mA sink) Can be used for gas alarm indication, or for status indication etc.
Status	CMOS unprotected. High Output level in the range 2.3V min to DVDD = 3.3V. (1 mA source) Low output level 0.75V max (4 mA sink)
Inputs	
Din0, Din1, Din2, Din3, Din4	Digital switch inputs, pull-up 120k to DVCC 3.3V. Driving it Low or connecting to ground G0 activates input. Pull-up resistance is decreased to 4..10k during read of input or jumper. Advantages are lower consumption most of the time the input/jumper is kept low and larger current for jumpers read in order to provide cleaning of the contact. Can be used to initiate calibration or to switch output range or to force output to predefined state. All depends on customer needs.
Din0, Din1, Din2, Din3, Din4	Digital switch inputs, pull-up 120k to DVCC 3.3V. Connecting to ground G0 activates input. Pull-up resistance is decreased to 4..10k during read of input or jumper. Advantages are lower consumption most of the time the input/jumper is kept low and larger current for jumpers read in order to provide cleaning current. They are the same as inputs on IDC connector. Can be used to initiate calibration or to switch output range or to force output to predefined state. All depends on customer needs.
I²C extension	
Contact SenseAir for information	Pull-up of SDA and SCL lines to 3.3V.

General PCB overview

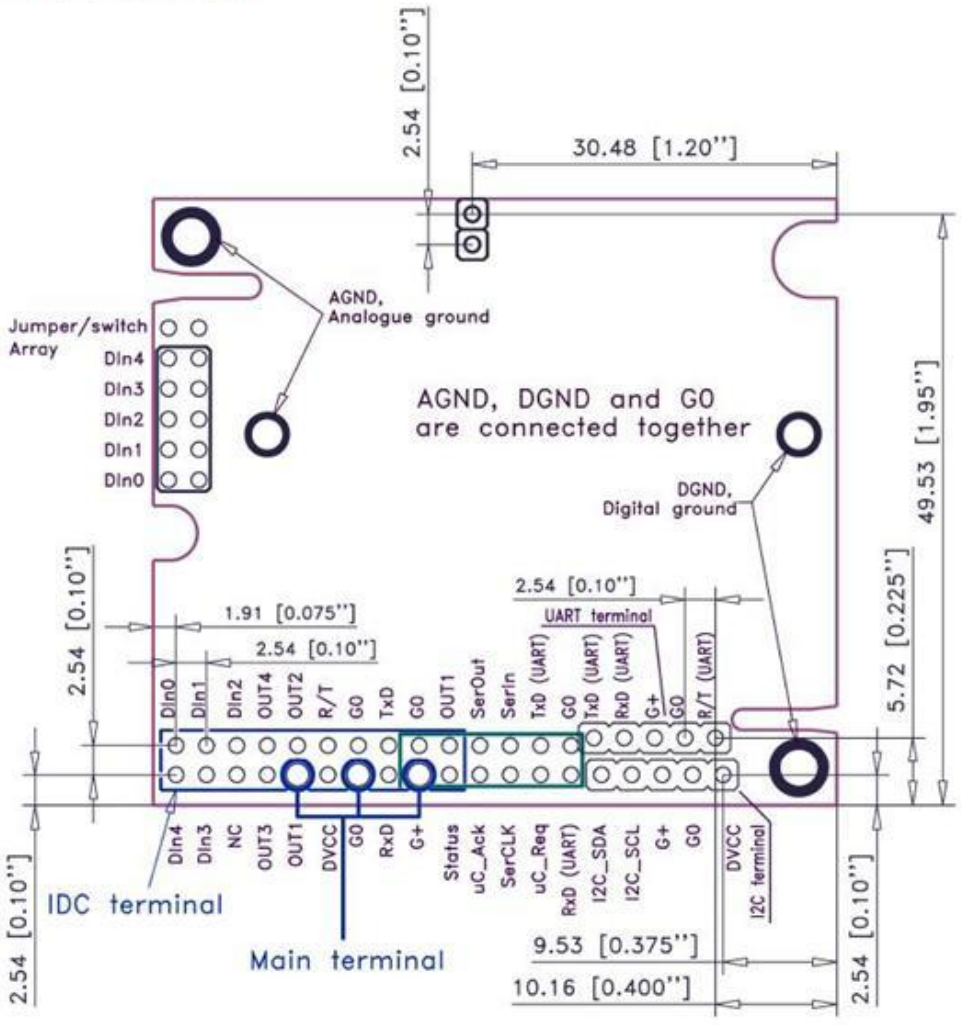


Figure 5- SENSEAIR K-30 CO_2 sensor PCB and Connection Overview
Source – SenseAirK-30 Data Sheet, Delsberg, Sweden, 2013 Aug.

2.2.3 GPS52D GPS receiver (Position Co)

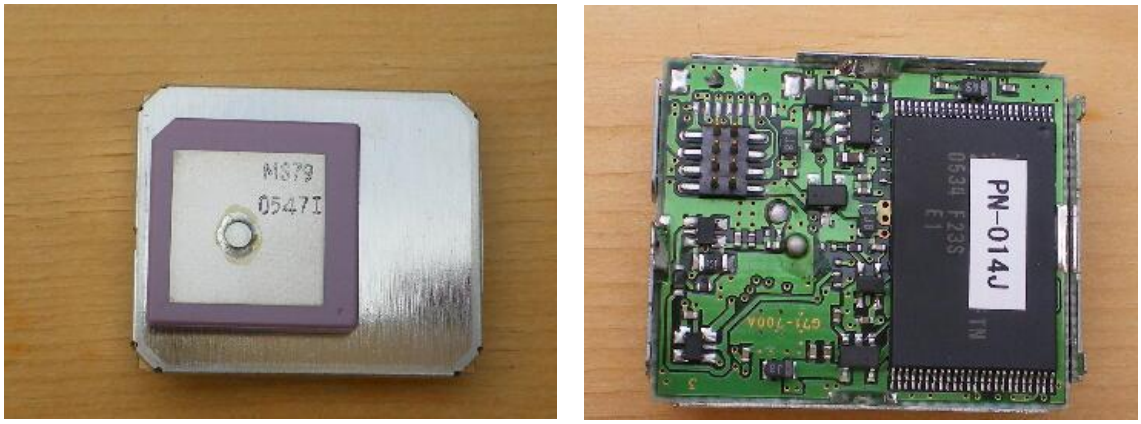


Figure 6 GPS52D GPS receiver

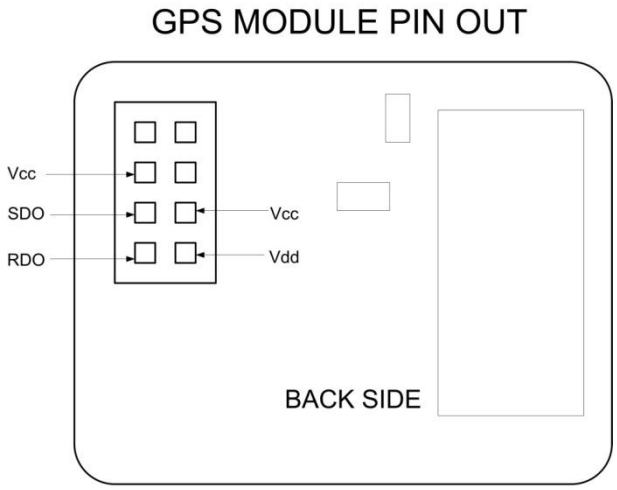


Figure 7 GPS Module Pin Out

This is a common GPS module available on market. The parameters are as follows. Cost is around 5-8USD

Parameters		*3-1 Description * 3-1
Receive mode		12-channel parallel
Receiving frequency		1575.42MHz ± 1MHz ， C/A 码 1575.42MHz ± 1MHz, C / A code
Sensitivity	Tracking	-142dBm -142dBm

	Capture	-137dBm -137dBm
Accuracy	Location	15m(二维)： GPS (SA=OFF ， PDOP ≤ 3) Than 15m (D): GPS (SA = OFF, PDOP ≤ 3)
	Speed	1m/s： GPS (SA=OFF ， PDOP ≤ 3) Better than 1m / s (rms): GPS (SA = OFF, PDOP ≤ 3)
News	Altitude	-500m~18000m -500m ~ 18000m
	Speed	1800km/h 以内 1800km / h or less
	Acceleration	2g 以内 2g or less
(TTFF) Positioning time (TTFF)	Cold start	70 seconds (typical values measured at room temperature)
	Warm start	38 seconds (typical values measured at room temperature)
	Hot Start	8 seconds (typical values measured at room temperature)
The smallest unit of measurement	Two-dimensional position	0.0001 分 0.0001 min
	Altitude	0.1m 0.1m
	Speed	0.01km/h ， 0.01 节 0.01km / h, 0.01 节
	Position	0.01 度 0.01 degrees
Data output frequency		1 time per second
Positioning		Two-dimensional and three-dimensional automatic switching
Low-power mode		Time setting and switch control
GPS Differential GPS		SBAS SBAS
Output data format		NMEA-0183 NMEA-0183 compatible
Power Supply	Normal mode	+3.1VDC ~ +3.6VDC +3.1 VDC ~ +3.6 VDC (measured at room temperature)
	Backup mode	+2.1 VDC ~ +3.6 VDC (measured at room temperature)
Current consumption	Normal mode	56mA ~75mA 56mA ~ 75mA (measured at room temperature)
	Backup mode	6 μ A (typical values measured at room temperature)
Operating temperature		- 30 ℃ ~ +80 ℃ - 30 ℃ ~ +80 ℃
Storage temperature		-40 ℃ -40 ℃ ~ +80℃ ~ +80 ℃
Dimensions		*3-10 30.8mm (W) × 25.8mm (D) × 9.7mm (H) including the antenna and shield * 3-10
Weight		12 grams or less, including the antenna and shield

2.2.4 LCD Panel HD44780

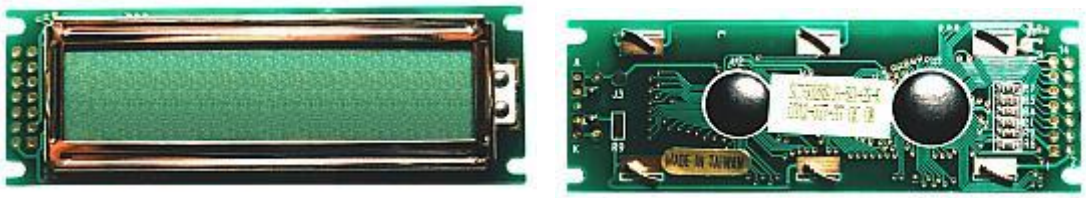


Figure 8 - LCD Panel HD44780

This is a liquid-crystal-display panel (LCD) for displaying time. This is a liquid crystal display made by SUNLIKE Co. in Taiwan. The display of two lines can be performed by 16 characters per line. This is a common LCD can find in market. Costs around 5USD. Low power consumption is the most attractive feature of this module. This modules parameters are as follows.

Absolute Maximum Ratings

(1) Electrical Absolute Ratings

Item	Symbol	Min.	Max.	Unit
Power Supply for Logic	V_{DD}-V_{SS}	-0.3	7.0	Volt
Power Supply for LCD	V_{DD}-V_O	-0.3	12.0	Volt
Input Voltage	V_I	-0.3	V_{DD}	Volt
LED Power Dissipation	P_{AD}	-	0.9	W
LED Forward current	I_{AF}	-	195	mA
LED Reverse Voltage	V_R	-	8	V

(2) Environmental Absolute Maximum Ratings

Item	Normal Temperature				Wide Temperature			
	Operating		Storage		Operating		Storage	
	Min,	Max.	Min,	Max.	Min,	Max.	Min,	Max.
Ambient Temperature	0℃	+50℃	-20℃	+70℃	-20℃	+70℃	-30℃	+80℃

Note 2 Ta≦50℃: 80% RH max
Ta>50℃: Absolute humidity must be lower than the humidity of 85%RH at 50℃
Note 3 Ta at -20℃will be<48hrs at 70℃will be <120hrs when humidity is higher than 70%.
Note 4 Background color changes slightly depending on ambient temperature. This phenomenon
is reversible.
Note 5 Ta≦70℃: 75RH max
Ta>70℃: absolute humidity must be lower than the humidity of 75%RH at 70℃
Note 6 Ta at -30℃will be <48hrs, at 80 ℃will be <120hrs when humidity is higher than 70%.

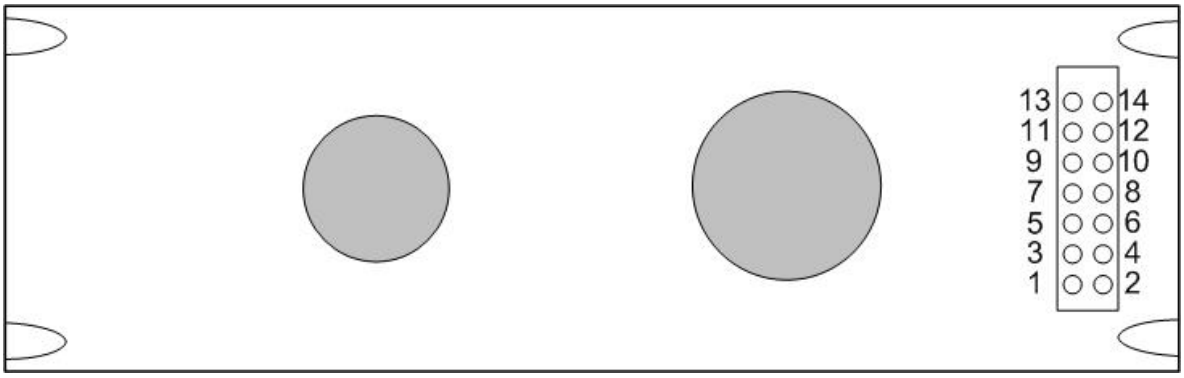


Figure 9 - Pinout of the LCD Panel HD44780

No	Symbol	Function
1	Vdd	5V
2	Vss	0V
3	Vo	Contrast Adj
4	RS	Register Select
5	R/W	Read/Write
6	E	Ebable Signal
7	DB0	Data BIT0
8	DB1	Data BIT1
9	DB2	Data BIT2
10	DB3	Data BIT3
11	DB4	Data BIT4
12	DB5	Data BIT5
13	DB6	Data BIT6
14	DB7	Data BIT7

Figure 10 - Pin Configuration of the LCD panel HD44780

2.2.5 Internal memory (EEPROM) ATMEL 24C1024

Figure 11 - Atmel 24C1024 EEPROM

This is an ATMEL company manufactured EEPROM. (Figure 3.1.5.1) Cost is around 5USD. The AT24C1024 provides 1,048,576 bits of serial electrically erasable and programmable read only memory (EEPROM) organized as 131,072 words of 8 bits each. The device's cascadable feature allows up to 2 devices to share a common 2-wire bus. The device is optimized for use in many industrial and commercial applications where lowpower and low-voltage operation are essential. The devices are available in spacesaving 8-lead PDIP, 8-lead EIAJ SOIC, 8-lead Leadless Array (LAP) and 8-ball dBGA packages. In addition, the entire family is available in 2.7V (2.7V to 5.5V) versions.

Features

• Low-voltage Operation
  – 2.7 (VCC = 2.7V to 5.5V)
• Internally Organized 131,072 x 8
• 2-wire Serial Interface
• Schmitt Triggers, Filtered Inputs for Noise Suppression
• Bi-directional Data Transfer Protocol
• 400 kHz (2.7V) and 1 MHz (5V) Clock Rate
• Write Protect Pin for Hardware and Software Data Protection
• 256-byte Page Write Mode (Partial Page Writes Allowed)
• Random and Sequential Read Modes
• Self-timed Write Cycle (5 ms Typical)
• High Reliability
  – Endurance: 100,000 Write Cycles/Page
  – Data Retention: 40 Years
• 8-lead PDIP, 8-lead EIAJ SOIC, 8-lead LAP and 8-ball dBGATM Packages

Description

Pin Configurations

Pin Name	Function
A1	Address Input
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect

Figure 12

Source – Atmel24c1024 Data Sheet, Atmel Corporation, 2013, July

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.25V
DC Output Current	5.0 mA

Figure 13

Source – Atmel24c1024 Data Sheet, Atmel Corporation, 2013, July

2.2.6 Arduino Ethernet Shield DEV-09026

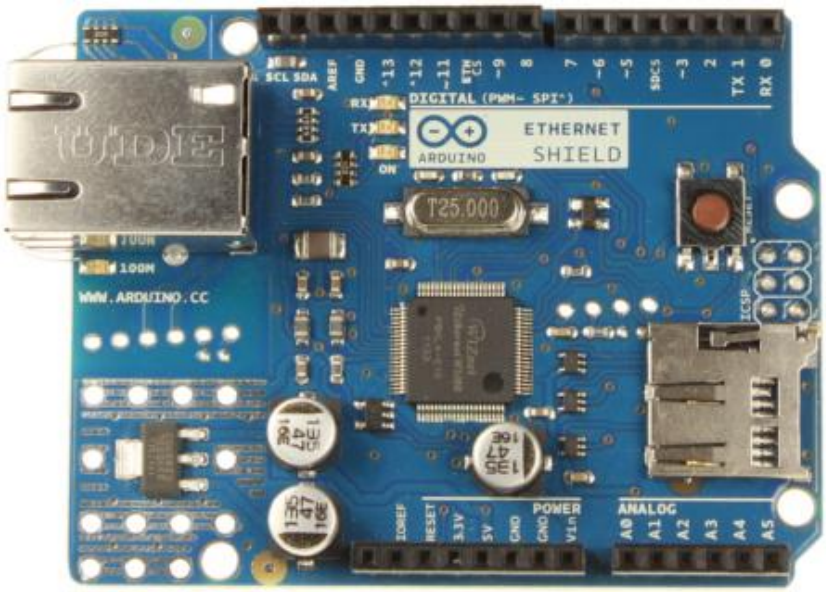


Figure 14 - Arduino Ethernet Shield DEV-09026

Arduino Ethernet shield is a module which made by arduino company by deploying ATMega328 microcontroller. This module has been utilized by this project to connect the system to the Ethernet and to the internet later on. Arduino programming language has been used to program this module to send readings to the KISSEL Server. The Wiznet W5100 Ethernet chip has been utilized by this module to communicate with Ethernet through RJ45 Interface. This chip capable both TCP and UDP protocols.

There are several numbers of indicator LEDs and it displays following information.

- PWR: indicates that the board and shield are powered
- LINK: indicates the presence of a network link and flashes when the shield transmits or receives data
- FULLD: indicates that the network connection is full duplex
- 100M: indicates the presence of a 100 Mb/s network connection (as opposed to 10 Mb/s)
- RX: flashes when the shield receives data
- TX: flashes when the shield sends data
- COLL: flashes when network collisions are detected

This shield transmits Data over the internet when it connects to Internet.

2.2.7 SIMCom 908 GSM/GPRS Module

This system has been attached a SIMCom908 GSM/GPRS Module as following figures

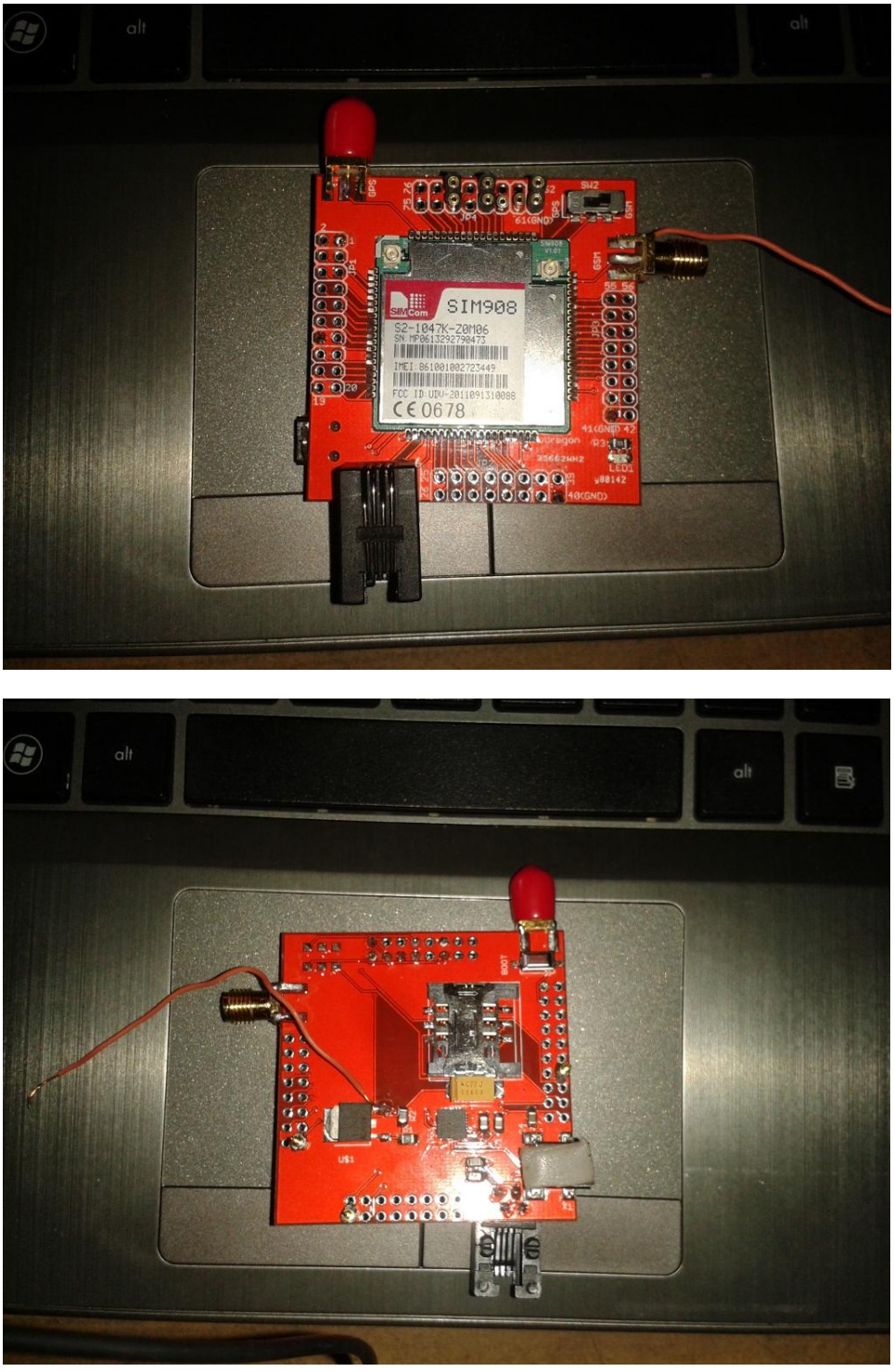


Figure 15- SIMCom 908 GSM/GPRS Module

SimCom908 is a Quad Band 850 / 900 / 1800 / 1900MHz module that supports with GPRS Multi-slot class 10, GPRS Mobile Station class B. This compliant to GSM phase 2 / 2+ - Class 4 (2W @ 850 / 900 MHz) and Class 1 (1W @ 1800 / 1900 MHz). Very

sophisticated and Dimensions are 30 x 30 x 3.2mm. This module can be controlled by AT commands. Supplied range of voltage is 3.2V – 4.8 V. This module is GPS capable but this project does not use that, uses more precise GPS module than it. This Module has been attached to the system, but its still on its configuration stage. It is expected to send data over the GPRS as a packet, which breaks the neediness of wired internet network, but improves the portability and mobility by improving its wireless communication capability. It is expected to complete the full attachment very soon to this system,

The data packet will send over the GSM network by developing and attaching this module to the system and will be able to use most of the places in the world where GSM signals are available. This improved the "Place Independency" and will be able to get relevant data more precisely where even internet or RJ45 interface is not available.

2.2.8 I²C BUS

Figure 16 - Official Logo for I²C Interface

AIRBOY uses I²C bus interface to communicate and store data in EEPROM. I²C (Inter-Integrated Circuit, generically referred to as "two-wire interface") is a multi-master serial single-ended computer bus invented by Philips that is used to attach low-speed peripherals to a motherboard, embedded system, or cellphone. I²C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted.

The I²C reference design has a 7-bit address space with 16 reserved addresses, so a maximum of 112 nodes can communicate on the same bus. Common I²C bus speeds are the 100 kbit/s standard mode and the 10 kbit/s low-speed mode, but arbitrarily low clock frequencies are also allowed. Recent revisions of I²C can host more nodes and run at faster speeds (400 kbit/s Fast mode, 1 Mbit/s Fast mode plus or Fm+, and 3.4 Mbit/s High Speed mode). Following figure 3.1.6.1 shows the conman connection of I²C bus.

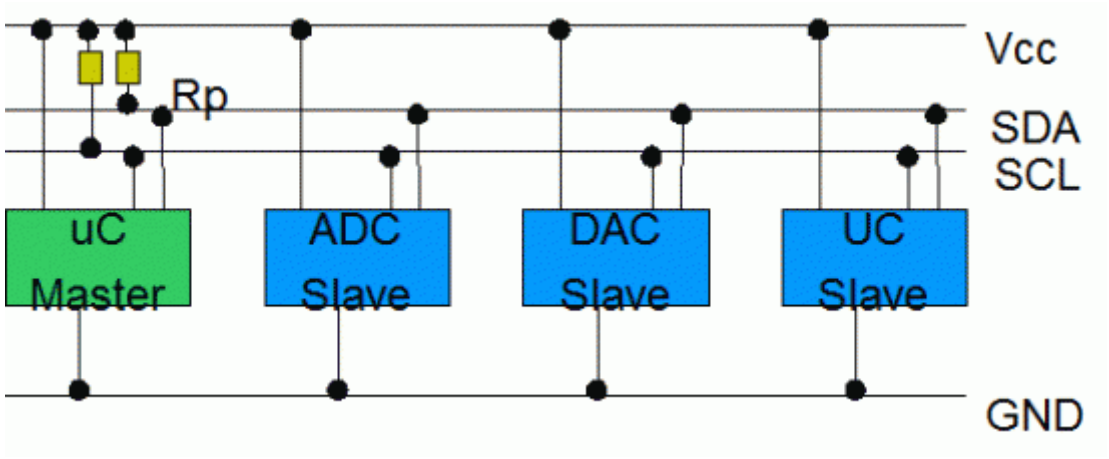


Figure 17 - I²C BUS Block Diagram

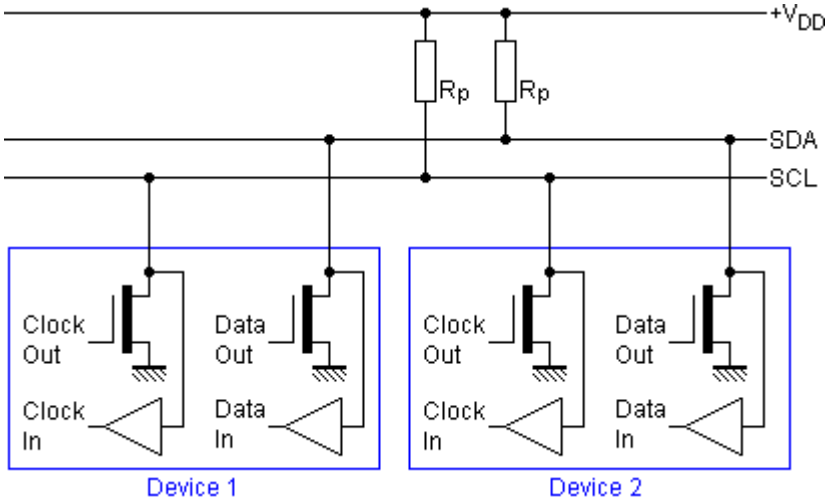


Figure 18 - I²C Interface Schematic

Figure 3.1.6.2. shows how to connect two or more devices to the bus in detail. When idle, the SDA and SCL lines are pulled up to the supply voltage with a pull-up resistor.

2.2.9 UART

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

AIRBOY uses this interface to communicate with GPS module and the external USB module when it downloads stored data from EEPROM

Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices.

Figure shows a 7o1 UART frame with 7 data bits, 1 parity and 1 stop bit, 11 bits in total. UART ’s character frame as follows.

Figure 19 - UART Character Framing

2.2.10 TCP/IP

In The system AIR-BOY, The Ethernet shield is use TCP/IP protocol to communicate with its server and send data to its database placed at KISSEL. TCP and IP were developed by a Department of Defense (DOD) research project to connect a number different networks designed by different vendors into a network of networks (the "Internet").

As with all other communications protocol, TCP/IP is composed of layers:

- **IP** - is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.
- **TCP** - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.
- **Sockets** - is a name given to the package of subroutines that provide access to TCP/IP on most systems.

2.2.11 MODBUS

MODBUS is an application layer messaging protocol which developed on 1979 to provide Client/Server communication between digital electronics which connected over different types of buses or networks. This positioned at level 7 at ISO/OSI 7 layer model, a elegent and simple structure which continued to grow, widely accessed by internet community at a reserved port 505 on the TCP/IP stack

In this system, K30 CO_2 Sensor communicates over the MODBUS protocol with the Arduino microcontroller as a request/reply module.

2.3 Advantages of Air Boy system and its Applications

2.3.1 Relatively small – easy to carry / transport

a. Data can be collected in wide range of locations (eg; Deep Ocean, top of the mountains, middle of deserts, Rain forests, urban areas etc..)
b. Comparison studies can be easily conducted because of its mobile activity (we can compare CO_2 concentrations in between Cities, countries and location to location
c. Useful for time series data collection (day to day, week to week, month to month and year to year etc..)

2.3.2 Relatively cheaper – therefore easily can be used in research activities rather than expensive methods. It is calculated that AIRBOY can manufactured around 150US$. It is found that a GPS+CO_2+TEMPERATURE measuring device is very hard to find in the market for any price.

2.3.3 User friendlyness – anyone can use, no technical knowledge need for operate. There are no special attention needed after its powered on..

2.3.4 Auto data saving capacity – (automatic data recording) It saves data automatically to internal EEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory) itself. Data capacity is 1 megabit for now. Saved data can be easily transferred to computer using USB interface.

2.3.5 Easy to data collection – we can fixed it required location or we can fixed in our vehicle when we traveling with required root. There is no maintenance required for this device.

2.3.6 Can be support different research and other activities- pollution monitoring, environmental studies, urban planning, Global warming research activities, green house gases monitoring programs etc, can deploy this device easily in low cost.

2.3.7 Light weight –

2.3.8 Durability- It can be combined with solar power, then it will keep long time period data collection activities

2.3.9 Mapping capability – Data associated with GPS locations. Therefore preparing of distribution maps will relatively easier than other CO_2 measuring devices. It is very easy to make a map with GOOGLE mapping system.

2.3.10 Accuracy and high efficiency

SENSEAIR K30 sensor is sensitive +-30ppm accuracy and can be used in 0-2000 ppm range.

2.3.11 Low Power Consumption

Its power consumption is 200mA when the device is full operational. In Low – Power GPS mode, its consumption is 180mA at 5V

3 AIRBOY Layout and its Outline

20 MHz Arduino Microcontroller gathers data from CO_2 and temperature sensors using I/O ports and in the same time from GPS through UART interface. Then it stored as raw data to the EEPROM using I²C bus protocol. Raw data of both sensors inputs as voltage levels. The CPU subjects inputs to ADC conversion and calculates the real values using Arduino programming language, while displaying calculated data through LCD panel using necessary transfer functions.

There are three inputs and three outputs to and from the CPU. The CO_2 sensor and the GPS module are interfaced using the serial port of the CPU. Each of these two devices is read through a multiplexer sequentially.

The GPS module repeatedly transmits GPS data as a character string. It is done at a rate of one set of sentences per second. These sentences are in the NMEA sentence [5] format. Between two sentence chunks, is a rest of about 500ms. The CPU looks for a pause of this length and captures the next sentence. This synchronization has to be done in the program because the GPS module runs in a different time base. These sentences in the chunk have to be parsed and necessary information has to be extracted in the CPU. Only two types of NMEA sentences are parsed in the CPU namely GPGGA and GPRMC. Following are two real example sentences.

$GPRMC,062906.499,V,0658.8112,N,08002.1135,E,005.0,089.0,100413,,,N*77
$GPGGA,062906.499,0658.8112,N,08002.1135,E,0,00,7.3,79.7,M,-91.7,M,,0000*40

The CPU has to check the validity of each sentence and parse only the valid sentences. The values extracted in the process are latitude, longitude, altitude, date, time and speed. Speed of travel is necessary to judge if the CO_2 readings were taken without moving or not.

The CO_2 sensor K-30 senses the air using the diffusion method. The chosen module converts the reading to a digital value using an onboard D/A converter. It then prepares the value for transmission in the requested format using ModBus[3] protocol. The values transmitted are in the range of 0ppm – 8000ppm. The CPU switches over from GPS module to the CO_2 sensor using a digital multiplexer and requests the CO_2 value from it. The sensor then sends the requested value.

The temperature sensor LM35 reads the temperature and outputs it as a highly linear analog voltage in the range of 0V to 5V. The CPU reads this voltage through one of its analog inputs.

The Atmel AVR CPU collects all these values and displays it on the 16x2 character Hitachi LCD module. The CPU sends the characters to the LCD module using six parallel digital lines. In addition, the CPU constructs a character string of minimum required length to neatly pack all the values. It then stores this packet in the EEPROM.

A 128KB ATMEL EEPROM memory is used to store the packets constructed in the CPU. Special memory algorithms were developed to utilize the memory at its best because the length of the packet may vary in changing versions and because the memory is accessed one page at a time. Difference in page size and our packet length leaves some bits unused. The developed algorithm uses these bits and makes the page operations transparent to the rest of the program.

This system utilizes the SPARKFUN Pro-Ethernet module to interface with LAN over the TCPIP prtocol, is used to communicate with the data upload PHP webpage. The CPU constructs an HTTP query string with the constructed packet having values read. It then makes a HTTP request to the PHP upload page through the LAN interface. The upload.php packet format as follows.

Upload.php?packet=xxxxxxxxxxxxxxxxxxxxx

"xxxx" represents the set of data which red and processed by Atmel CPU. An example data packet of location as follows.

Upload.php?packet=$GPRMC,062906.499,V,0658.8112,N,08002.1135,E,005.0,089.0,100413,,,N*77$GPGGA,062906.499,0658.8112,N,08002.1135,E,0,00,7.3,79.7,M,-91.7,M,,0000*40

There is no acknowledgement which getting from server side. By receiving the data set/packet, the program processes it and separates each values of CO_2 level, Temperature, Altitude, Time and Position. Then it writes to MYSQL database on kissel server.

- **HTTP/1.0 GET request**

```
http://kissel.base.ibaraki.jp/airboy/upload.php?packet=31382f30332f37370031383a31363a303000a0a0a0a0b0b0b0b0c0c0d0e0f0f0
```

The microcontroller makes the above GET request with the acquired data composed as a single hexadecimal query string parameter.

index.php page reads data in MYSQL database and integrates it with color code and displays over the GOOGLE MAPS API. Figure III shows the entire system architecture and figure IV shows the actually implemented prototype.

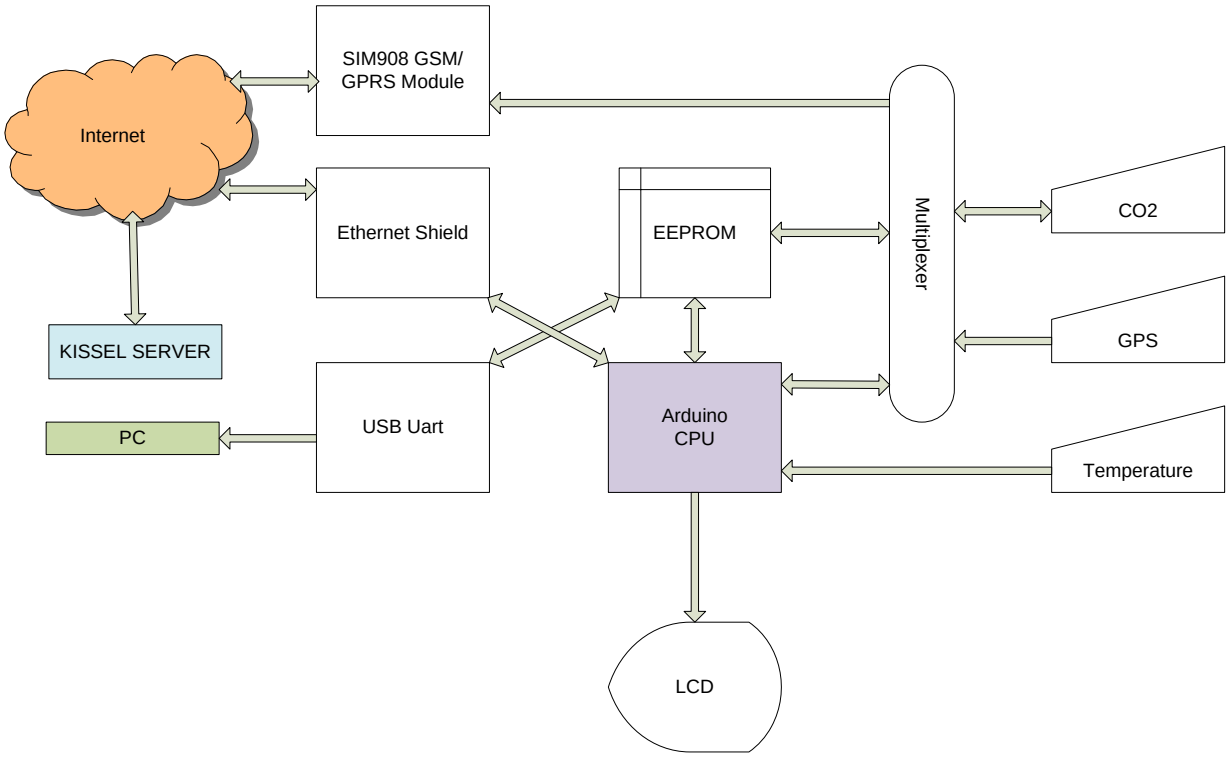


Figure 20 - AIRBOY System Architecture

SENSEAIR- K30 infrared CO_2 sensor has been deployed to sense the CO_2 level. CO_2 level determine by PPM (Particles per million). It communicates with Arduino CPU, over the MODBUS protocol. This K30 CO_2 Module measures CO_2 level and send it to the CPU through a multiplexer to compare with Temperature over the time which obtained from GPS sensor and Temperature sensor.LM35 well known IC module has been used for sense the air temperature. It senses temperature in Celsius and it will easily can support the user to take his decisions because the CO_2 level and Temperature level has a sort of mutual relationship inside the atmosphere, mostly an ascending similarity. The CO_2Engine® K30 is basically maintenance free in normal environments because of the built-in self-correcting algorithm. Basically the AIR-BOY is a maintenance free device which needs only the power from externally.

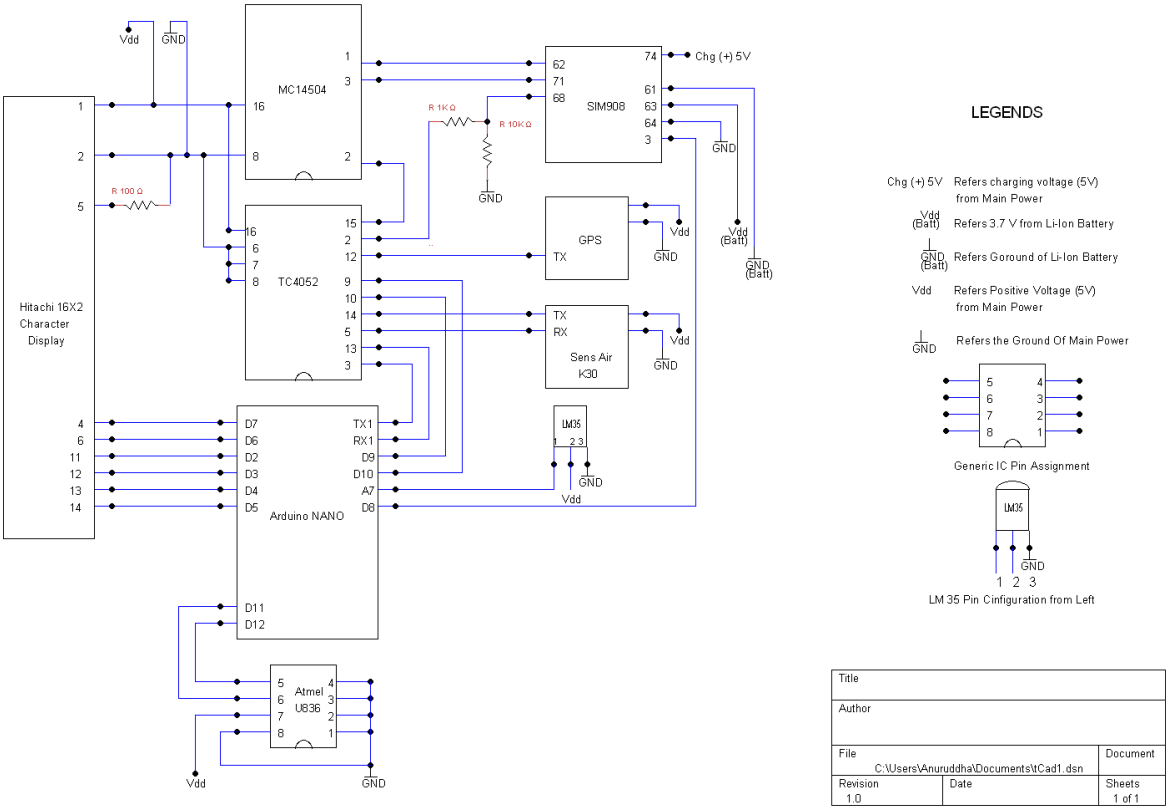


Figure 21 Circuit Diagram

Co2.h,i2c.h,at24c.h,lcd.h header filers are in-house developed or customized header files. These file has been created or altered to suite of needs of the AIR-BOY project.

3.1 Co2.h

This file consist only data structures regarding the sensors and saved data of EEPROM. There are four structures are defined inside, those are,
GPRMC, GPGGA, GPS, EEPROM_RECORD

3.1.1 GPRMC- the incoming GPRMC sentence from GPS unit is parsed and stores inside the data structure for further processing. From this sentence, several parameters are taken. This GPRMC sentence is known as the "Recommended Minimum" sentence, and the most common sentence received by most of domestic GPS devices such as Navigations, smart phone applications etc... Following data are taken and stored among the incoming data set of GPRMC sentence for further processing.

- Latitude
- Longitude
- Speed
- Date
- Time

The format of the GPRMC sentence as follows

$GPRMC,HHMMSS.SS,A,DDMM.MMM,N,DDDMM.MMM,W,Z.Z,Y.Y,DDMMYY,D.D,V,M,NS*CC<CR><LF>

This one sentence contains nearly everything a normal GPS application needs those are

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the GPS position
A	Status (A = valid, V = invalid)
DDMM.MMM	Latitude in degrees, minutes, and decimal minutes
N	Latitude location (N = North latitude, S = South latitude)
DDDMM.MMM	Longitude in degrees, minutes, and decimal minutes
W	Longitude location (E = East longitude, W = West longitude)
Z.Z	Ground speed, in knots
Y.Y	Track made good, reference to true north
DDMMYY	UTC date of position fix in day, month, and year
D.D	Magnetic Variation, in degrees
V	Variation sense (E = East, W = West)
M	Mode indicator Variable length valid character field type with the first two characters currently defined. • First character indicates the use of GPS satellites • Second character indicates the use of GLONASS satellites If another satellite system is added to the standard, the mode indicator will be extended to three characters. New satellite systems shall always be added on the right, so the order of characters in the Mode Indicator

	is: GPS, GLONASS, other satellite systems in the future. The characters shall take one of the following values: • N = No fix. Satellite system not used in position fix, or fix not valid • A = Autonomous. Satellite system used in non-differential mode in position fix • D = Differential. Satellite system used in differential mode in position fix • P = Precise. Satellite system used in precision mode. Precision mode is defined as no deliberate degradation (such as Selective Availability) and higher resolution code (P-code) is used to compute position fix. • R = Real Time Kinematic. Satellite system used in RTK mode with fixed integers • F = Float RTK. Satellite system used in real time kinematic mode with floating integers • E = Estimated (dead reckoning) mode • M = Manual input mode • S = Simulator mode The mode indicator shall not be a null field.
NS	Navigational status; options are: • S = Safe • C = Caution • U = Unsafe • V = Not valid for navigation
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

GPGGA – the incoming GPGGA sentence from GPS unit is parsed and stores inside the data structure. AIRBOY uses GPGGA , NMEA data format to receive its data from GPS receiver, –AIR BOY– System uses $GPGGA GPS Fix format for the system with WGS-84

datum and following data are taken and stored among the incoming data set of GPGGA sentence for further processing.

3.1.2 GPGGA Data Format

$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh

1 = UTC of Position , **2** = Latitude ,**3**= N or S
4= Longitude, **5** = E or W **6** = GPS quality indicator (0=invalid; 1=GPS fix; 2=Diff. GPS fix)
7 = Number of satellites in use [not those in view]
8= Horizontal dilution of position, **9**= Antenna altitude above/below mean sea level (geoid), **10** = Meters (Antenna height unit),**11** = Geoidal separation (Diff. between WGS-84 earth ellipsoid and mean sea level.=geoid is below WGS-84 ellipsoid), **12** = Meters (Units of geoidal separation), **13** = Age in seconds since last update from diff. reference station, **14**= Diff. reference station ID#, **15** = Checksum. **AIR BOY**- is used 1,2,3,4,5,6,9 data for its calculations.

In early stage, Data has been gathered around HITACHI city ibaraki ken Japan around 15 Km. There were 300 data values were found in EEPROM. Following **Figure 5.0** shows some data sets which downloaded from EEPROM

```
090436344187N140384900E0,33d,7d
090736347417N140389129E1,2a5,7a
090936351547N140389168E0,2a5,77
091436353843N140398275E1,29d,79
091836364702N140405750E0,2b5,76
092236367214N140400125E0,2a9,73
092536361734N140392321E1,29d,70
093336347518N140382851E1,2f5,73
093836343149N140386029E1,29d,71
094136344133N140384975E1,295,71
```

Received Data in EEPROM

Last 6 digits separated by two commas are CO_2 and Temperature levels respectively. Following figures show GPS, CO_2 and Temperature measurement when the device is operational. The detailed explanation of a data set is described in following Figure 5.1.

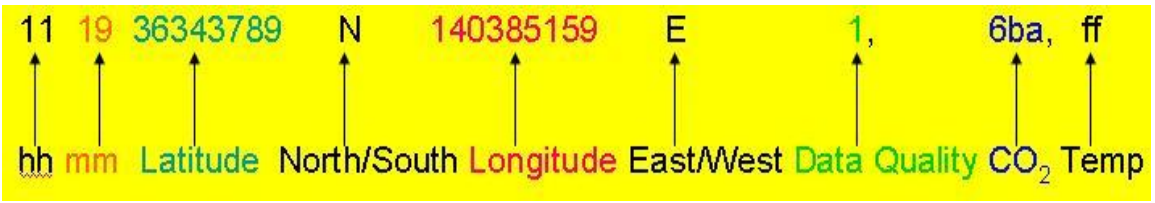


Figure 22 - Data Packet

The system Uses following parameters which extracts from the GPGGA sentence for its calculations.

- Altitude
- Fixed quality
- Number of satellite

The example of GPGGA sentence as follows
$GPGGA,001038.00,3334.2313457,N,11211.0576940,W,2,04,5.4,354.682,M,-26.574,M,7.0,0138*79

The syntax of GPGGA sentence as follows.
$GPGGA,HHMMSS.SS,DDMM.MMMMM,K,DDDMM.MMMMM,L,N,QQ,PP.P,AAAA.AA,M,±XX.XX,M,SSS,RRRR*CC<CR><LF>

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the GPS position
A	Status (A = valid, V = invalid)
DDMM.MMM	Latitude in degrees, minutes, and decimal minutes
N	Latitude location (N = North latitude, S = South latitude)
DDDMM.MMM	Longitude in degrees, minutes, and decimal minutes
W	Longitude location (E = East longitude, W = West longitude)
Z.Z	Ground speed, in knots
Y.Y	Track made good, reference to true north
DDMMYY	UTC date of position fix in day, month, and year
D.D	Magnetic Variation, in degrees
V	Variation sense (E = East, W = West)
M	Mode indicator Variable length valid character field type with the first two characters currently defined. • First character indicates the use of GPS satellites

	• Second character indicates the use of GLONASS satellites If another satellite system is added to the standard, the mode indicator will be extended to three characters. New satellite systems shall always be added on the right, so the order of characters in the Mode Indicator is: GPS, GLONASS, other satellite systems in the future. The characters shall take one of the following values: • N = No fix. Satellite system not used in position fix, or fix not valid • A = Autonomous. Satellite system used in non-differential mode in position fix • D = Differential. Satellite system used in differential mode in position fix • P = Precise. Satellite system used in precision mode. Precision mode is defined as no deliberate degradation (such as Selective Availability) and higher resolution code (P-code) is used to compute position fix. • R = Real Time Kinematic. Satellite system used in RTK mode with fixed integers • F = Float RTK. Satellite system used in real time kinematic mode with floating integers • E = Estimated (dead reckoning) mode • M = Manual input mode • S = Simulator mode The mode indicator shall not be a null field.
NS	Navigational status; options are: • S = Safe • C = Caution • U = Unsafe • V = Not valid for navigation
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

GPS – this data structure utilized to extract the relevant GPS data from above two structures.
EEPROM_RECORD – this data structure has been utilized for to prepare the data to be written to the eeprom this data consists the GPS data + CO_2 + TEMP data.

3.1.3 I2C.h

This file is used to facilitate i2c bus communication using two pins of the micon. This contents functions to work the two pins to create i2c protocol messages. These messages are init, start condition, stop condition, ack, no ack, reset, read byte, write byte. There are separate functions for each of these messages which make the two pins work.

3.1.4 At24c.h

This file uses the i2c.h file and contains the functions nessesary to communicate with an external atmel eeprom memory. There are eight functions included in this file.

init (scl pin, SDA pin, memory size, page size)

this function initializes the external eeprom device using supplied parameters. this at24c.h library allows us to read and write to the device disregarding the page size of the device. This library keeps track of the page size and changes the page accordingly to the read or writes operation being performed.

```
at24c.init(11,12,131072,256);
```

this initializes the device whose scl pin and sda pin are connected to the micon with data pins 11 and 12 respectively. This device used by the project has a memory size of 131072 bytes (128Kb) and a page size of 256 bytes.

3.1.5 read_byte (address)

This function reads a byte at the givan address and returns it as an 8 bit unsigned integer

```
uint8_t a = read_bye(0x0a)
```

This function reads a byte from the address 0x0a of eeprom and returns it to the variable a.

3.1.6 read_bytes(address,buffer,length)

this function reads a byte stream of a givan length and stores them in to the buffer provided as a parameter. It also returns the buffer.

```
char buffer[128];
   at24c.read_bytes(0x0a,buffer,128);
```

this function reads 128 bytes starting from address 0x0a and stores the bytes in to the char array provided.

3.1.7 search_byte(bits)

This function calls the overloaded function search_byte(bits,0,1) . This is useful to easily find the first occurence of a particular byte value. it returns the address as a 32bit integer and returns -1 if the byte value is not found.

3.1.8 search_byte(bits,start address)

this function works as the previous function but the search begins at the start address.
it returns the address as a 32bit integer and returns -1 if the byte value is not found.

3.1.9 search_byte(bits,start address,n)

this function works as the previous function but finds n^{th} occurrence of the given byte value. it returns the address as a 32bit integer and returns -1 if the byte value is not found.

3.1.10 write_byte(address,bits)

this function writes a given byte value at the given address. it returns -1 if the address id bigger than the memory size. it returns 0 if the write was successfully verified and returns -2 if the write verification is failed.

3.1.11 write_bytes(address, buffer,length)

This function writes a byte stream of a given length starting from the address provided. this function handles all the page operations. it returns -1 if the address + length is bigger than the memory size. it returns 0 after completion.

3.1.12 lcd.h

This libraray leverages the work of an open source 3rd party LCD library and is completely re-written and coustomized to adopt the needs of this project.

3.1.13 init(rs,en,d7,d6,d5,d4)

this function initializes an LCD device connected to the number of pins passed in as parameters.

3.1.14 print(character)

This function prints a single character at the current cursor position.

3.1.15 println(character)

prints a single character in a new line.

3.1.16 Print(text)

this function prints a string starting at the current cursor position.

3.1.17 printlk(text)

prints a string, starting at a new line.

3.1.18 print(int)

this function prints an integer starting at the current cursor position.

3.1.19 println(int)

this function prints an integer starting at a new line

3.1.20 print(long)

prints a long integer starting at the current cursor position.

3.1.21 println(long)

prints a long integer starting at a new line

3.1.22 cursor(int 8_x, int8_y)

this function position the cursor at a given x,y coordination of the LCD

3.1.23 cursor_on()

This makes the cursor visible

3.1.24 cursor_off()

This makes the cursor invisible

3.1.25 clear

This function clears the display.

3.1.26 getGPS()

3.2 EEPROM saving Structure

The data stream that saves to the EEPROM memory has arranged as a packet called “er” derived from the EEPROM_RECORD data type. “er” consist of the GPS, Temperature and CO_2 data. “er” gets updated once an every cycle of the main loop and always bear latest relevant data of sensors. Two types of delimiters are used to separate data packets and end of line. The packet separating delimiter is used as 0x1E with reference of the ASCII table as displays in below, means the “record separator “ (30^{th} decimal position). 0x03 has been used to denote “end of text” according to the ASCII table on its 3^{rd} position in decimal. The storing method of data in EEPROM as follows.

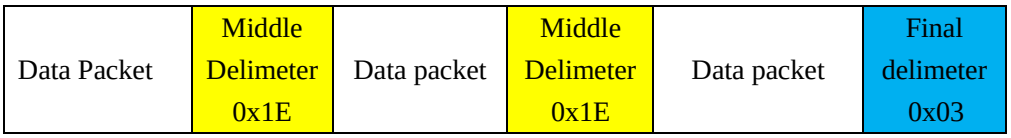


Figure 23 - storing method of data in EEPROM

3.3 ASCII Table

In following figure, the ASCII character table can be found. The 30th Decimal point represents the record separator and the 3rd Decimal point represents the final delimiter of a data set.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Figure 24 - storing method of data in EEPROM

Tomas Vilda, ASCII Table,[Online],

Available: http://tvilda.stilius.net/ascii/ascii.php , 14 Dec 2012

4 The Program

4.1 Program Structure

The structure of the program, elaborated as flow charts in later pages. The main program has been divided to two charts.

4.1.18 Initialization

4.1.19 Main loop

It can be found other functions, stated according to its complexity, which included to header files which made by especially for this project as

4.1.20 The GetGPS process in main loop, explains receiving and storing method of two types of GPS sentences.

4.1.21 Dividing process of GPS data in to data sets and inserting delimiters as necessary

4.1.22 Writing process to EEPROM

4.1.23 Finding the next registers to be written on the EEPROM

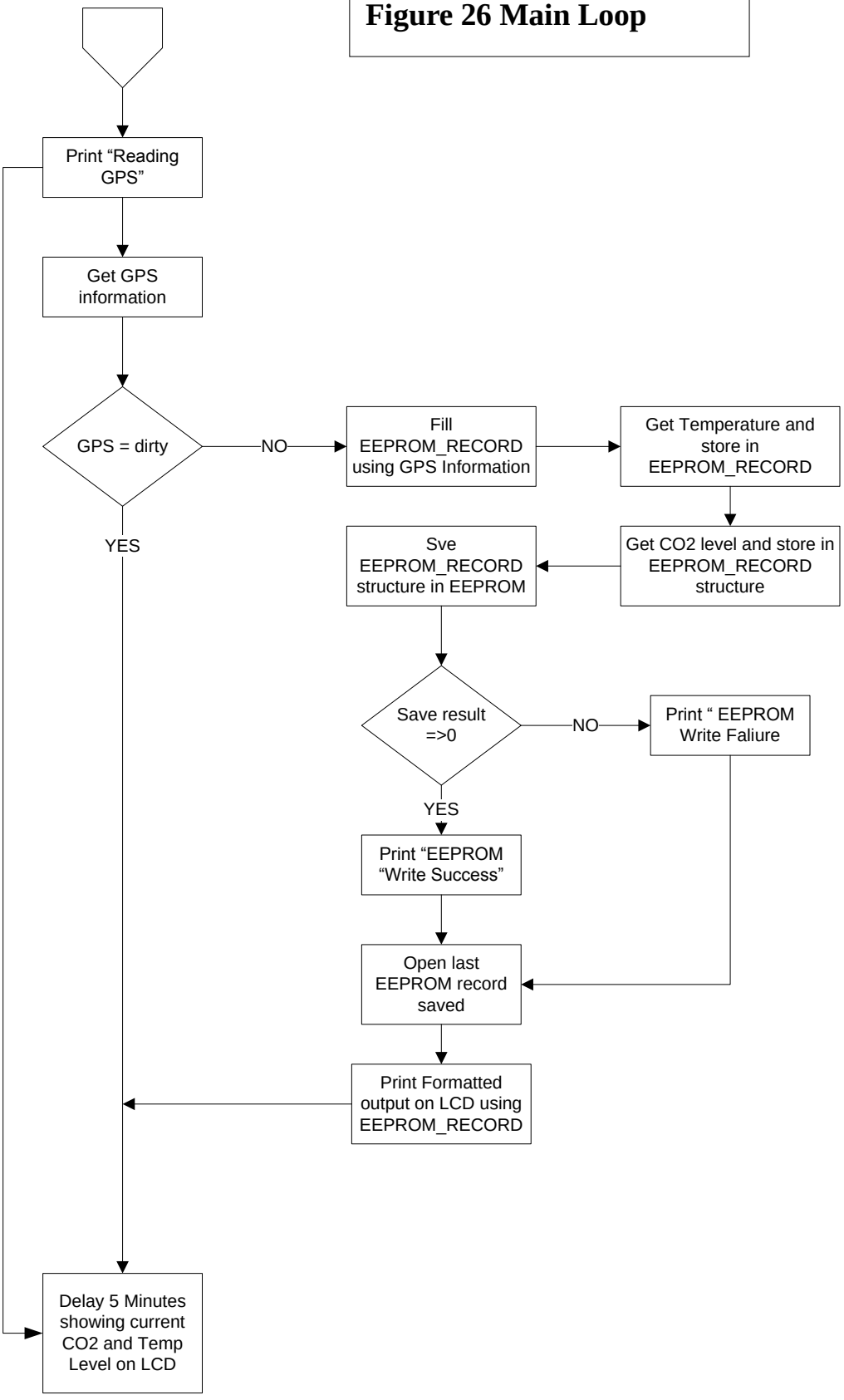
4.2 Web Interface Programming

PHP is used for creating the web interface at Kissel server. The Program is as follows. There are two PHP files are used for whole web constructions so far.

4.2.1 **Index.php** - Written for display the data base and its values and display the google map according to the respective coordination

4.2.2 **upload.php –** This file parses the quary string received from the microcontroller. Then it decompose the qary string in to data and stores the values in to the database table.

Figure 26 Main Loop



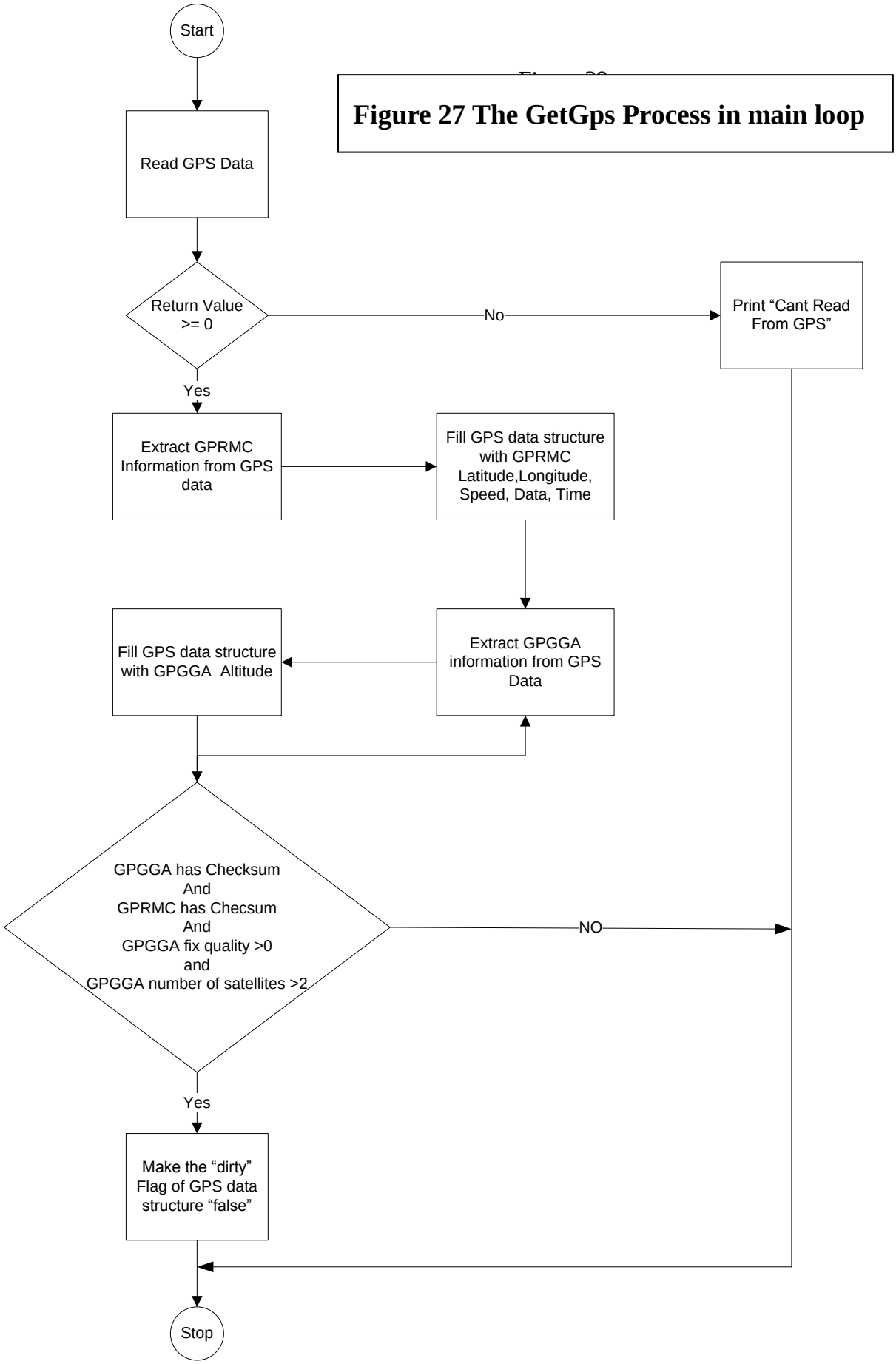


Figure 27 The GetGps Process in main loop

Figure 28 : Dividing Process of GPS data in to data sets and inserting delimeters as nessasary

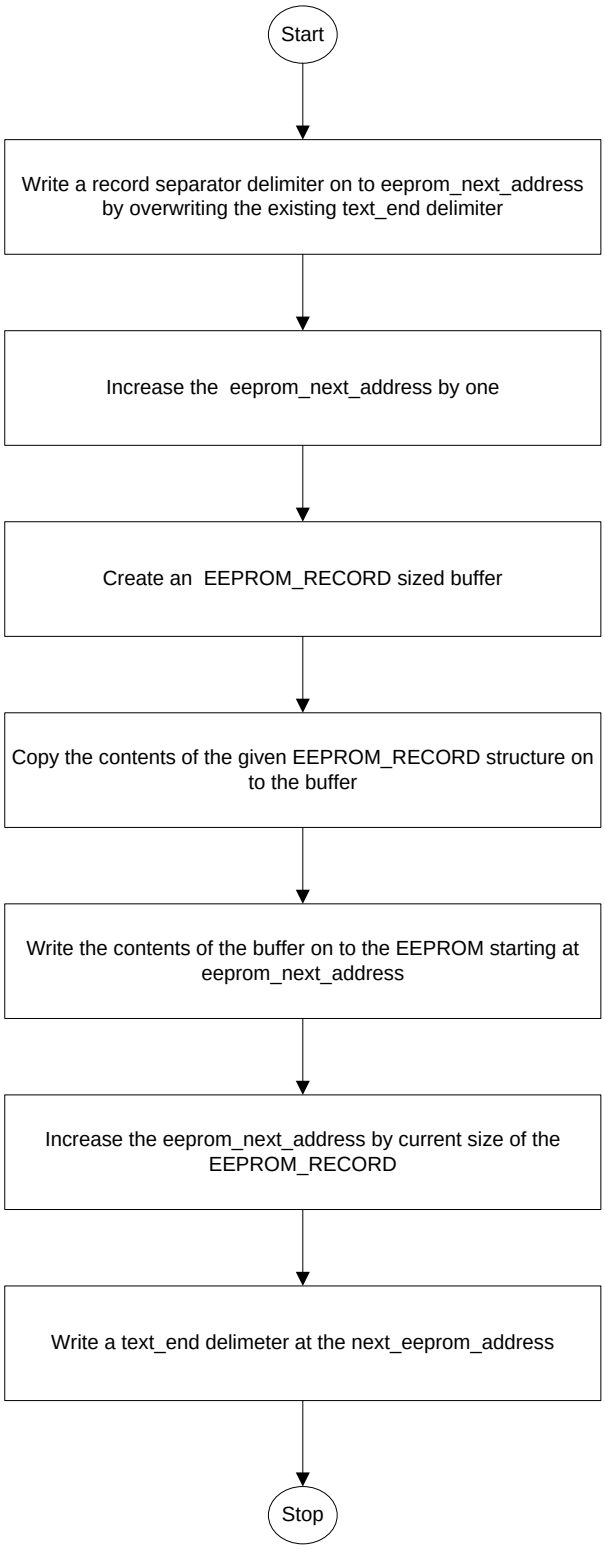


Figure 29 : Writing Process to EEPROM

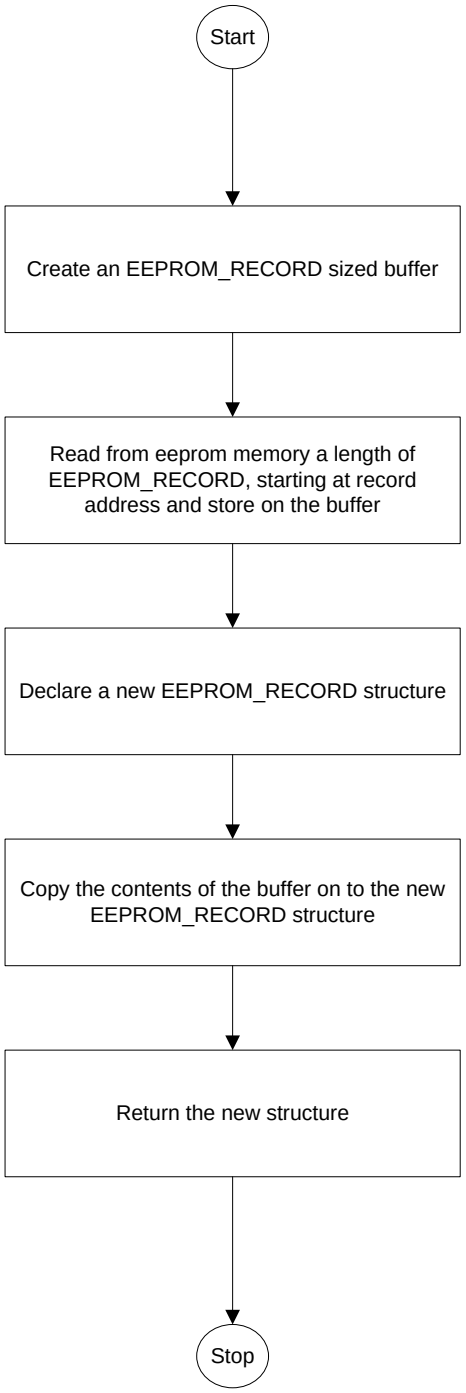
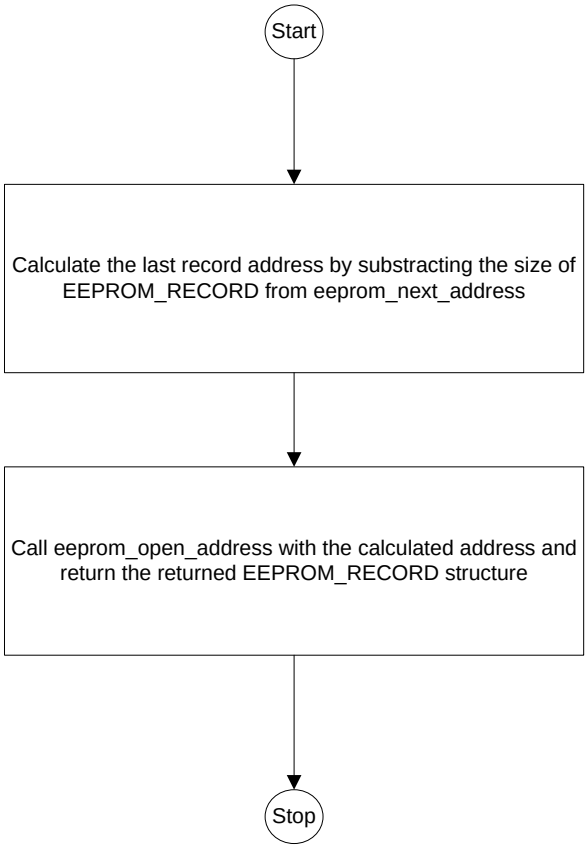


Figure 30: Finding the next registers to be written on the EEPROM



5. AIRBOY Assembly.

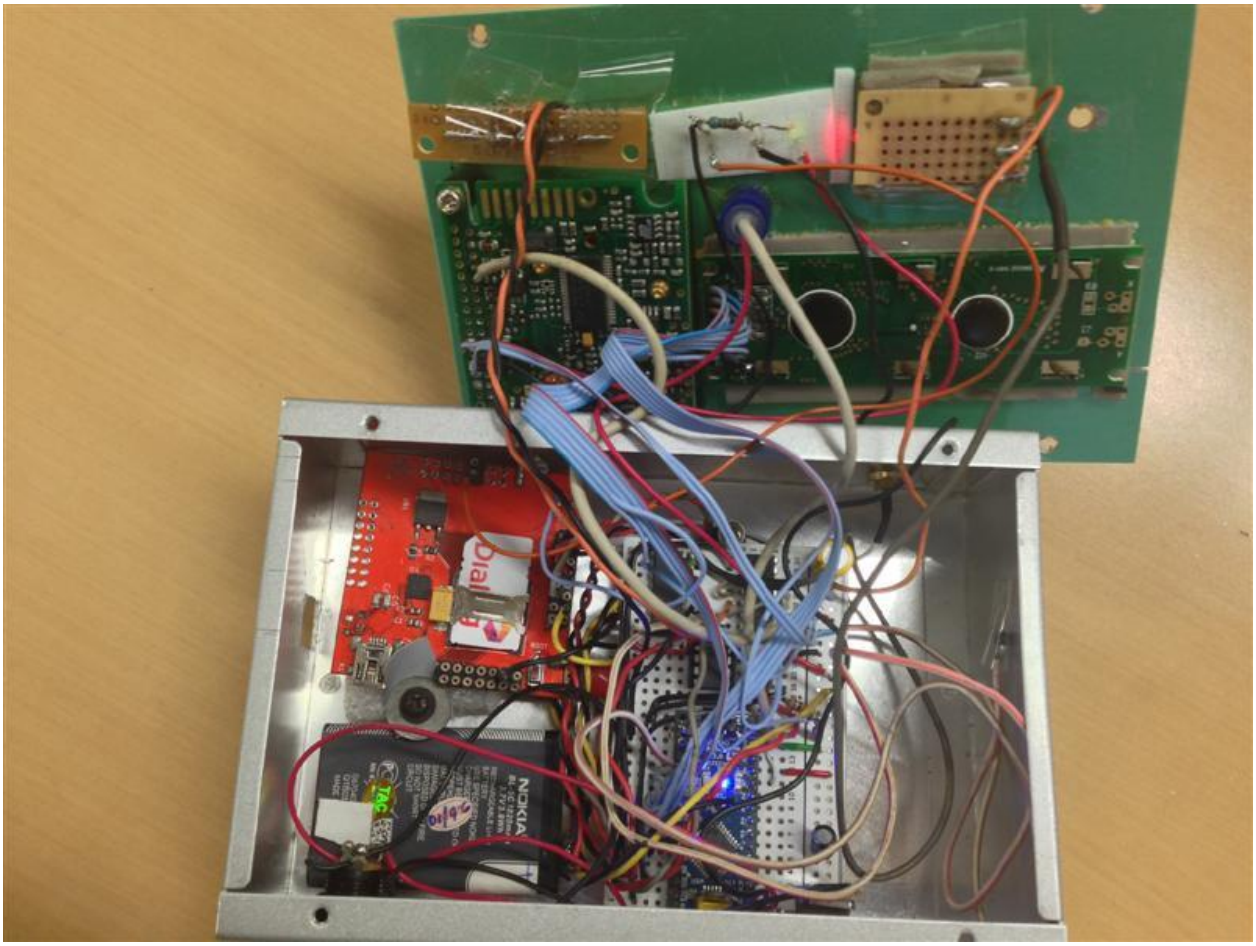


Figure 31 - Actual Components setout

Figure 32 - Actual Prototype

5.1 Calibration

The calibration phase has done using a real CO_2 and Temperature devices. The real Device was " EASY SENSE" a United Kingdom made device manufactured by Data Harvest Inc. The Calibration setup was done as follows (Figure 4.4.1)

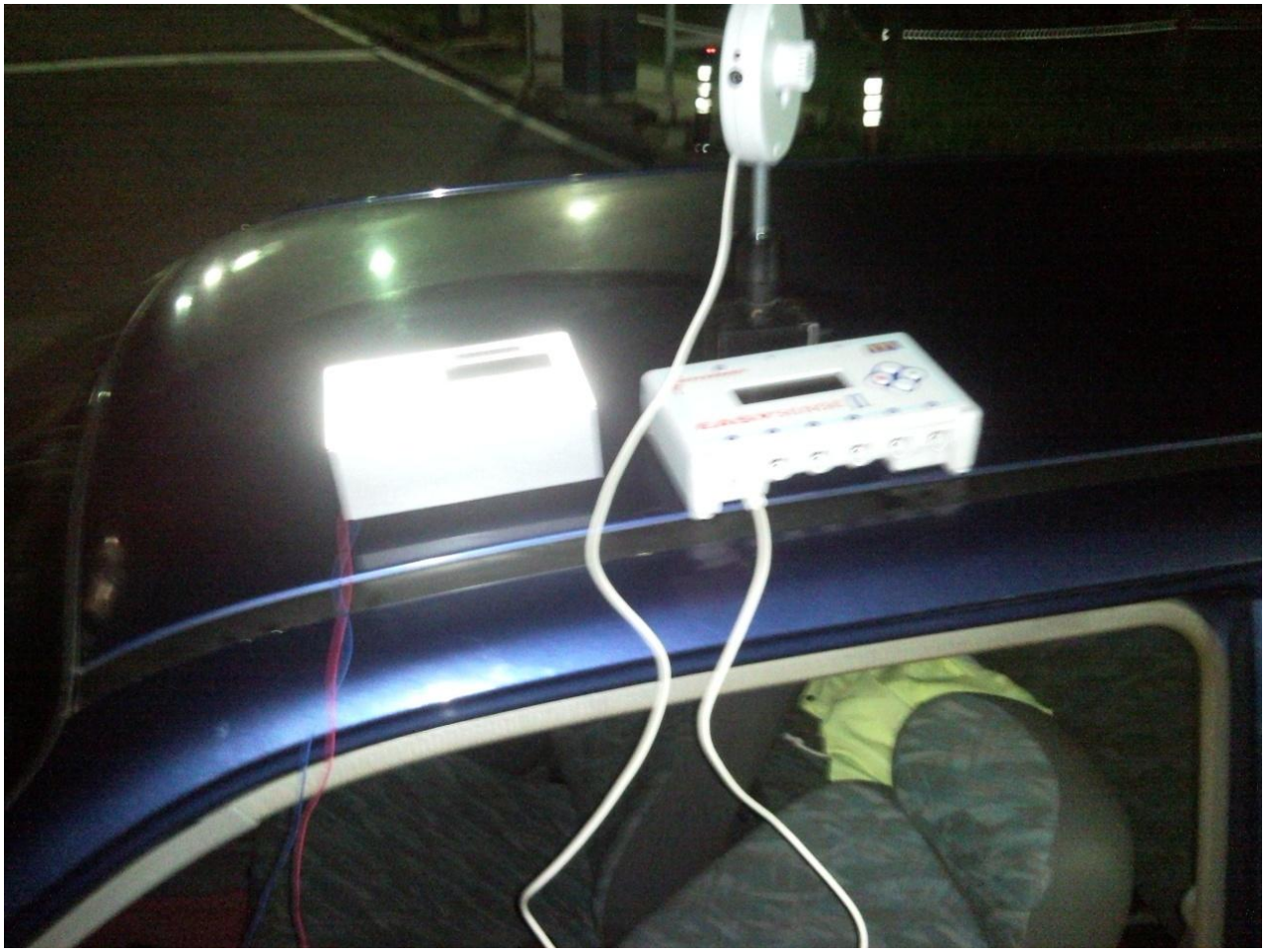


Figure 33- Calibration of AIRBOY

5.2 Data Gathering and Analyzing

In the early stage of development of this system, 40 data values has been selected out of 300 values to propagated manually using Google Maps. The map is shows in following figure 5.2

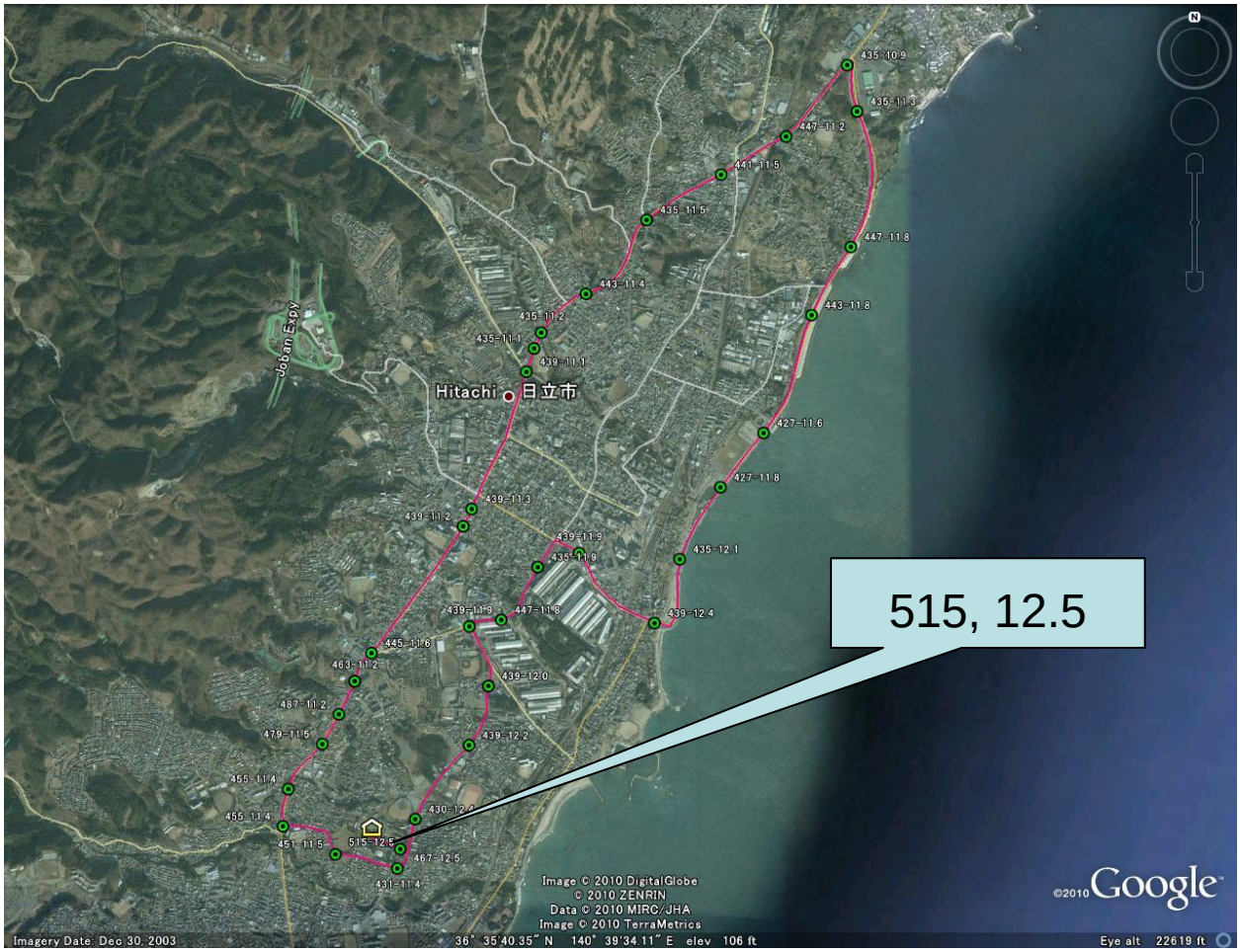


Figure 34 - Mapping around Hitachi city

The magnified data value implies the particular point consist 515 ppm CO_2 value and 12.5 C of temperature. The time that gatherd data was 18.04 – 18.41 hours and the whole distance was 15.3 Km. Green Dots are GPS points and red lone is the path that has used to check

. A graph has plotted using whole 300 data values by comparing CO_2 Vs Temperature, is shown in Figure 47

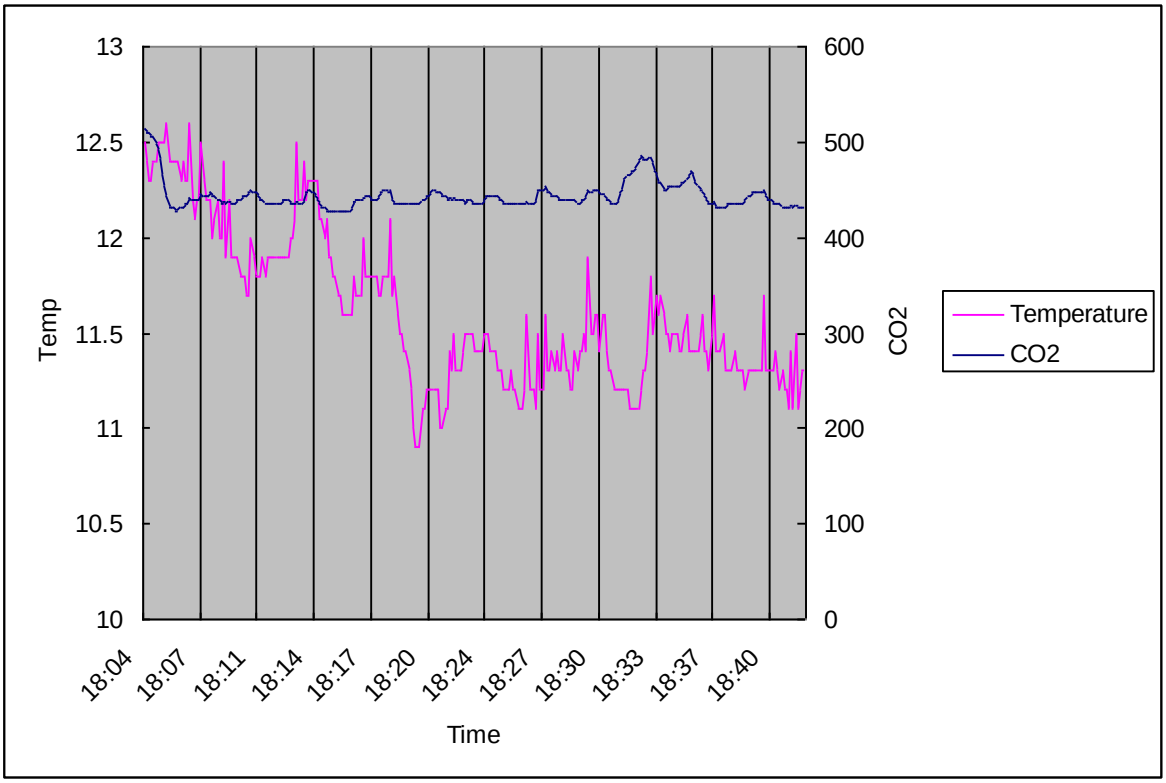


Figure 35 - CO_2 and Temperature Level Over the time (weather factor is not removed)

. This system can be installed in some of major metropolis in the world such as Tokyo, New Delhi, New York etc… This allows having actual image of CO_2 emission and its density over the temperature, time and its respective altitudes. It is very hard to find this type of service in internet. This will be more accurate free and will be able to use anyone who needs real CO_2 readings, comparatively inaccurate and very expensive remote sensing satellite data. Several trial have been conducted internationally to verify the functionality of the device. Following images has been generated in recent trial

The device uses an ATMEL AVR microcontroller with a program written in C using the Arduino platform. The task of the program is to validate the data received from the input devices and upload the data to the web server periodically. In addition, it also saves the uploaded data inside a separate EEPROM memory to be later downloaded to the PC if needed. The microcontroller uses a LAN device with the TCP/IP protocol to send the http request to the data acquisition PHP page.

5.3 Presenting Data on a Colour Coded Map

The received data from devices across the globe is validated in the server side and stored in the MySQL database. The users visit the map webpage and the map is displayed using a colour coded CO_2 layer on top of a Google Map. The map is then updated periodically without refreshing the page using AJAX[1]. The site will be hosted in the KISSEL[2] server of Ibaraki University Kansei Mathematics Lab. Refer figure I for colour codes.

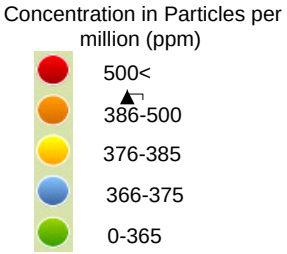


Figure 36 - CO_2 Monitoring Color Code

5.4 Trial Observations

Figure 37 – Trial Monitoring – Japan

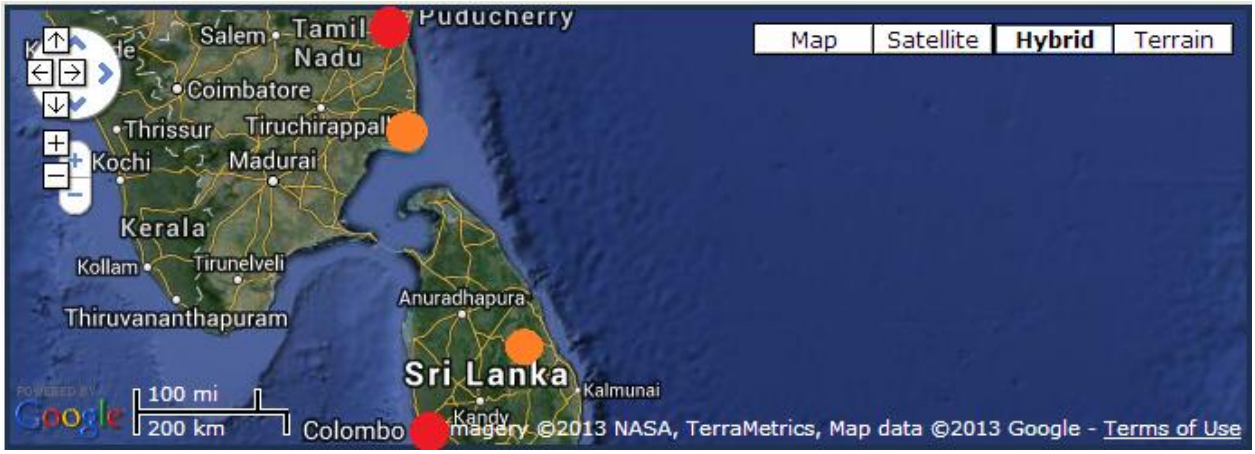
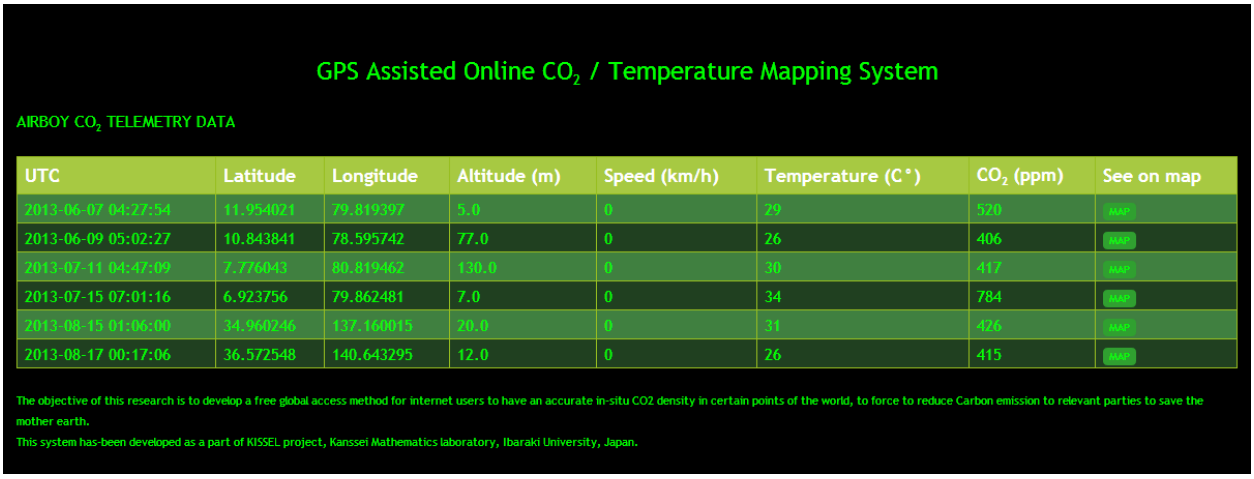


Figure 38 – Trial Monitoring India and Srilanka

0

System gathered CO_2 data in Srilanka, Japan and India as a Trial monitoring session. It showed a certain results and successfully transmitted it in to a database on the web. a Screenshot of the webpage as follows. By clicking on the "See On Map" button, user can see the map of respective data. The color coding part is still to be developed.



GPS Assisted Online CO_2 / Temperature Mapping System

AIRBOY CO_2 TELEMETRY DATA

UTC	Latitude	Longitude	Altitude (m)	Speed (km/h)	Temperature (C°)	CO_2 (ppm)	See on map
2013-06-07 04:27:54	11.954021	79.819397	5.0	0	29	520	MAP
2013-06-09 05:02:27	10.843841	78.595742	77.0	0	26	406	MAP
2013-07-11 04:47:09	7.776043	80.819462	130.0	0	30	417	MAP
2013-07-15 07:01:16	6.923756	79.862481	7.0	0	34	784	MAP
2013-08-15 01:06:00	34.960246	137.160015	20.0	0	31	426	MAP
2013-08-17 00:17:06	36.572548	140.643295	12.0	0	26	415	MAP

The objective of this research is to develop a free global access method for internet users to have an accurate in-situ CO2 density in certain points of the world, to force to reduce Carbon emission to relevant parties to save the mother earth.
This system has-been developed as a part of KISSEL project, Kanssei Mathematics laboratory, Ibaraki University, Japan.

Figure 39 - AIR-BOY Telemetry Data web

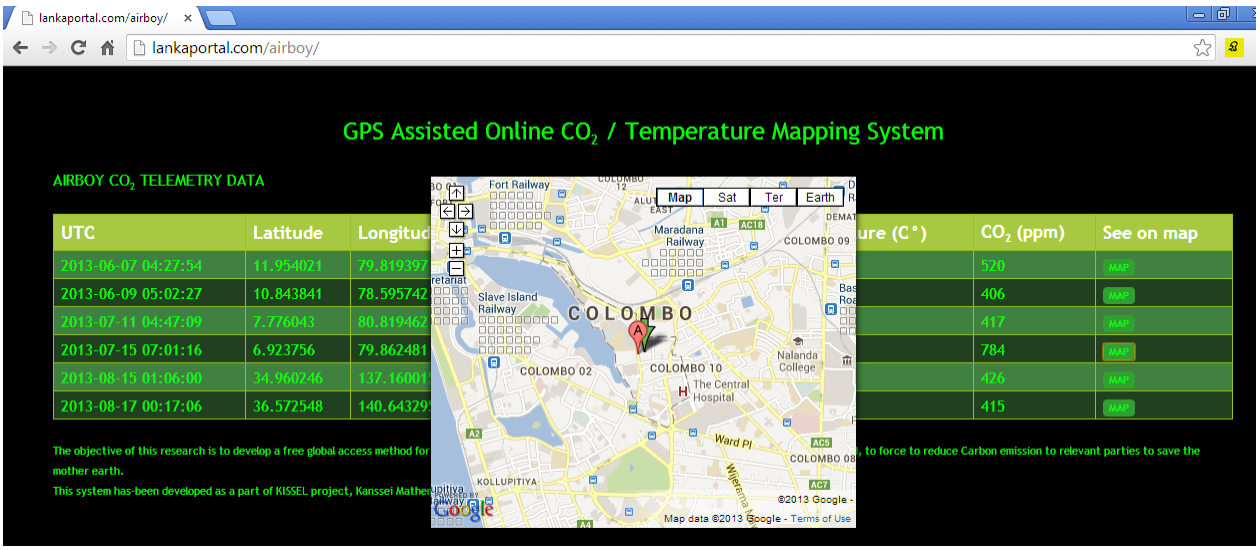


Figure 40 - Map of a Location of the monitoring

In India, The system gathered CO_2 data in Puducherry City and Tiruchirappali City. In Puducherry, it is observed that the CO_2 level is slightly higher in middle of the city limit and observed 14 hours as on its mean of entire reading. In time of whole observation, the device kept on the same place and didn't move to anywhere. In this observation, the Device has been gathered data successfully including GPS coordination and transmitted to its server over the internet successfully.

In Tiruchirapalli has relatively lower amount of CO_2, observed 8 hours on outer border of

the city. Calculation has done on its mean of entire reading. In this area, it is observed that the Level of CO_2 is slightly lower than previous place, Puducherry. In Tiruchirapalli, the windy surroundings has been observed during the monitoring time. In time of whole observation, the device kept on the same place and didn't move to anywhere. In this observation, the Device has been gathered data successfully including GPS coordination and transmitted to its server over the internet successfully.

In srilanka, The city called Bakamuna, In Polonnaruwa District has been observed for 48 hours.It is observed that the Level Of CO_2 is slightly lower than Observed next City in srilanka, the capital Colombo. Colombo was the most contaminated City on whole observation, which has higher CO_2 level. In Colombo, it is observed CO_2 data for 48 hours of time. In time of whole observation, the device kept on the same place and didn't move to anywhere. In this observation, the Device has been gathered data successfully including GPS coordination and transmitted to its server over the internet successfully.

In japan, CO_2 level of Okazaki-shi, Aichi-ken, was observed during the trial. It is found that the CO_2 level around the area is higher than the next place of observation, Ibaraki University, Hitachi City in Ibaraki prefecture. In time of whole observation, the device kept on the same place and didn't move to anywhere. In this observation, the Device has been gathered data successfully including GPS coordination and transmitted to its server over the internet successfully.

In whole observations, Thiruchirapalli enjoys the lowest rate of CO_2 among six cities of three countries, 406 mean PPM. Colombo displayed the highest rate of CO_2 is 784 mean ppm. However, there is an error to be corrected. The time factor should be removed from these data because in day time, the Oxygen concentration in atmosphere is high and in night time the CO_2 rate is higher than O2. A fair technique should be introduced to remove this error.

5.5 Commissioning

The fully developed system has been commissioned at one of highly polluted highland city in Srilanka called KANDY. Data gathered in most polluted area of the city, in front of Central Bus Stand and the Train station. Both Bus and Train services are Diesel Driven in srilanka. This City called Kandy is a one of important world heritage site according to the UNESCO.

Figure 41 Location of monitoring (Goods Shed Bus Stand - Kandy)

There are hundreds of ancient temples are around the city with ancient paintings and carvings. It has been reported of fading colors, and damaged itself of ancient paintings. It is reported that ancient cave paints of Altamira in spain also damaged by the same incident.

"During the 1960s and 1970s, the paintings were being damaged by the carbon dioxide in the breath of the large number of visitors. Altamira was completely closed to the public in 1977, and reopened to limited access in 1982." newworldencyclopedia.org. (2013) *ESA: Altamira Cave:* [Online]. Available from: http://www.newworldencyclopedia.org/entry/Altamira_(cave) [Accessed 3rd january 2014]

Higher density of carbon dioxide might cause this and it is very important to measure the

density of green house gases using this kind of online portable monitoring system. There are many lung diseases and respiratory problems are also reported around the Kandy city. After a brief monitoring within 36 hours, the data was depicted; almost whole day the density of CO_2 keeps same comparing with Colombo

2014-02-05 12:05:49	7.289202	80.630157	497.0	0	27	373	MAP
2014-02-05 12:04:20	7.289202	80.630157	497.0	0	27	371	MAP
2014-02-05 12:02:52	7.289198	80.630150				370	MAP
2014-02-05 12:01:24	7.289198	80.630150				369	MAP
2014-02-05 11:59:56	7.289192	80.630150				371	MAP
2014-02-05 11:58:20	7.289186	80.630157				376	MAP
2014-02-05 11:56:51	7.289170	80.630165				373	MAP
2014-02-05 11:55:22	7.289160	80.630180				371	MAP
2014-02-05 11:53:51	7.289135	80.630180				370	MAP
2014-02-05 11:52:19	7.289118	80.630188				367	MAP
2014-02-05 11:50:50	7.289148	80.630180				366	MAP
2014-02-05 11:49:21	7.289166	80.630165				364	MAP
2014-02-05 11:47:53	7.289175	80.630173				361	MAP
2014-02-05 11:46:24	7.289175	80.630173				362	MAP
2014-02-05 11:44:55	7.289173	80.630165	506.0	0	27	363	MAP
2014-02-05 11:43:26	7.289162	80.630150	508.0	0	27	365	MAP
2014-02-05 11:41:58	7.289132	80.630142	512.0	0	28	365	MAP
2014-02-05 11:40:29	7.289163	80.630165	506.0	0	27	364	MAP
2014-02-05 11:39:00	7.289166	80.630157	506.0	0	27	366	MAP
2014-02-05 11:37:31	7.289175	80.630157	502.0	0	29	368	MAP
2014-02-05 11:36:02	7.289170	80.630150	503.0	0	28	368	MAP
2014-02-05 11:34:33	7.289165	80.630142	503.0	0	28	369	MAP
2014-02-05 11:33:04	7.289165	80.630142	503.0	0	29	369	MAP
2014-02-05 11:31:35	7.289165	80.630142	503.0	0	28	370	MAP

Figure 42 Commissioning and Data gathering

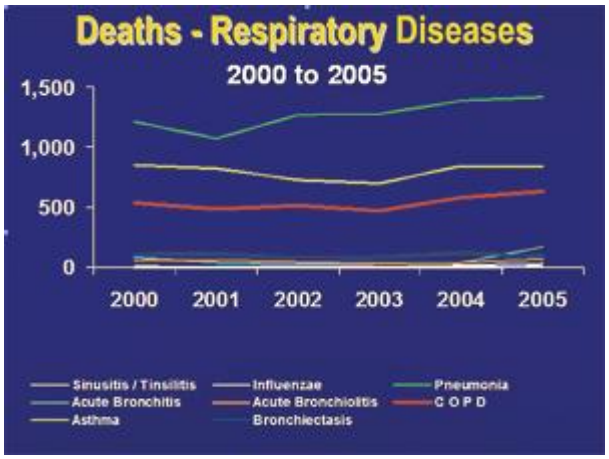


Figure 43- Respiratory Diseased Deaths in Kandy

(Statistics from Statistical Division of Ministry Of Health Srilanka, 2013, Nov)

Monitoring session started at +5. 30 GMT in 2014.02.14 at Kandy City. The first data received GPRS by the server, at +5. 33GMT, after three minutes, turning power on. This time usually take the system to initialize GPS and GSM modules in the start. After receiving first data, approximately one reading had been received from the server wirelessly for 36 hours. It is observed that data varied from 74ppm within 17 hours. Again it comes to approximately same 365ppm in 2014.02.05 at 8.58GMT, after 24 hours of smallest reading. There was only 4ppm difference monitored in the 24 hours observation loop. Refer Figure 51, 52, 53.

UTC	Latitude	Longitude	Altitude (m)	Speed (km/h)	Temperature (C °)	CO_2 (ppm)	See on map
2014-02-04 09:00:25	7.289198	80.630058	485.0	0	32	362	MAP
2014-02-04 08:58:56	7.289172	80.630173	499.0	2	33	361	MAP
2014-02-04 08:57:27	7.289268	80.630280	498.0	3	30	363	MAP
2014-02-04 08:55:58	7.289218	80.630379	516.0	3	30	363	MAP
2014-02-04 08:54:28	7.289299	80.631050	552.0	8	30	369	MAP

Figure 44– The lowest ppm (361ppm)of carbon dioxide of the 36 hours observation session

2014-02-05 02:07:14	7.289142	80.630203	495.0	0	19	428	MAP
2014-02-05 02:05:45	7.289130	80.630196	495.0	0	19	432	MAP
2014-02-05 02:04:16	7.289122	80.630211	499.0	0	20	432	MAP
2014-02-05 02:02:47	7.289180	80.630180	493.0	0	22	433	MAP
2014-02-05 02:01:18	7.289176	80.630173	492.0	0	19	435	MAP
2014-02-05 01:59:49	7.289173	80.630173	491.0	0	22	435	MAP
2014-02-05 01:58:20	7.289173	80.630173	492.0	0	18	434	MAP
2014-02-05							

Figure 45– The Highest ppm (435ppm)of carbon dioxide of the 36 hours observation session

09:04:33	7.289115	80.630173	525.0	0	30	365	MAP
2014-02-05 09:03:04	7.289120	80.630180	517.0	2	30	368	MAP
2014-02-05 09:01:33	7.289058	80.630280	543.0	0	29	367	MAP
2014-02-05 09:00:04	7.289120	80.630272	533.0	0	30	365	MAP
2014-02-05 08:58:35	7.289142	80.630264	519.0	0	30	365	MAP
2014-02-05 08:57:06	7.289122	80.630280	525.0	2	30	366	MAP

Figure 46– CO_2 level comes again to The approximately same (only 4ppm difference monitored) lowest ppm, (365ppm) after exactly 24 hours from firstly monitored lowest ppm.

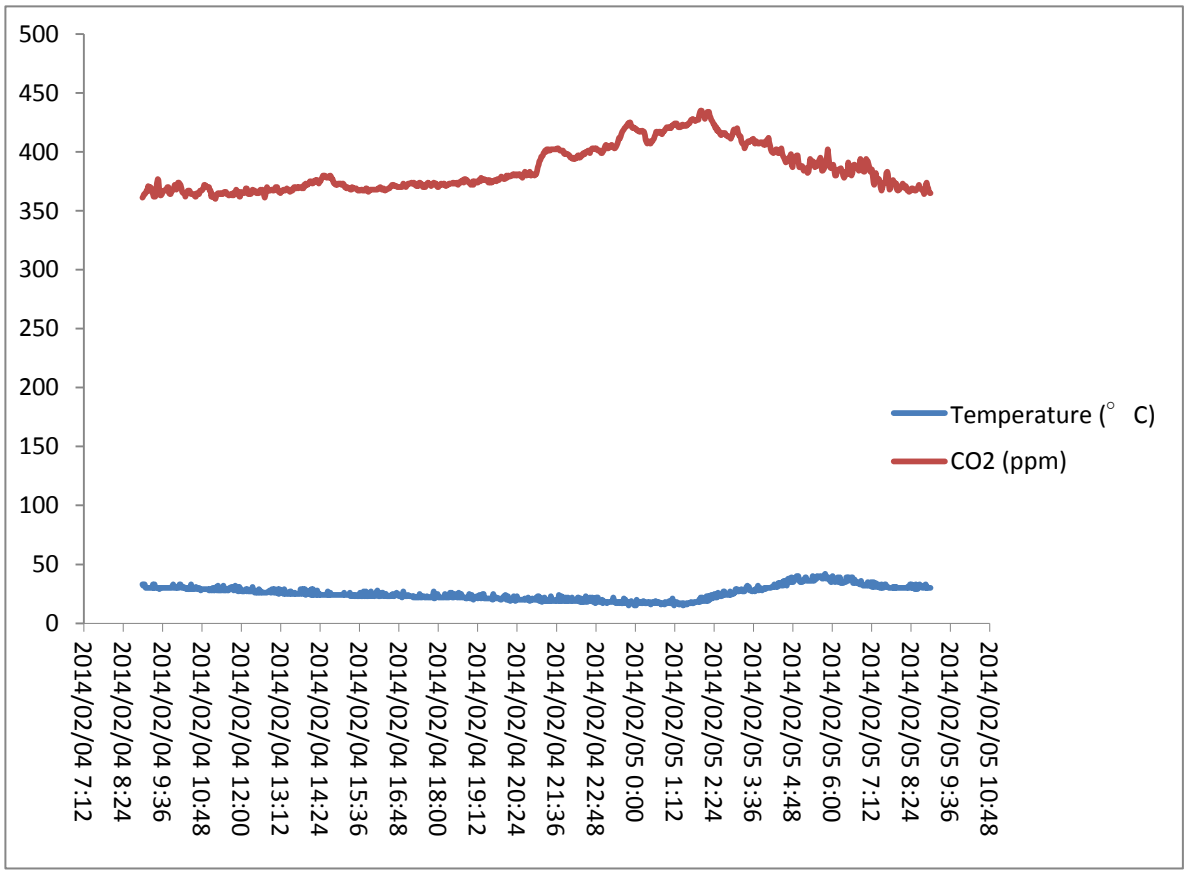


Figure 47– Chart for the 24 hour CO_2 level loop (Time in UTC)

In the Figure 54, It can be seen more precisely that the CO_2 level comes back to the approximately same level after 24 hours, just varies with 4ppm.

This time in UTC and should be add +5.30 hours to get local time. The highest peak if CO_2 shows in 7.30 – 8.00 AM local time, the busiest time in the day of the City. Thousands people around there in this rush time and can be seen the most vehicular traffic of the day. This place is very near to the train and bus station. Big spikes of CO_2 emissions can be observed within the later hours.

5.6 Conclusion

This environmental case study in KANDY CITY has done to verify the functionality of AIRBOY system. This system successfully could take data, and send it to its server over the GSM network. This is is accessible to public over the internet..

This GPS Assisted Online CO_2-Temperature Mapping System prototype has following advantages and disadvantages over the remote sensing satellite imaging technique.

Advantages

1. Anyone can access data in free of charge
2. Real time monitoring
3. Device is light-weight and can save data and can upload when it connect to the internet
4. Very cheap and economical

Disadvantages

1. Cannot transfer data without an internet connection
2. It is very hard to make a smooth CO_2 flowing image as in remote sensing technique, because CO_2 level changes very sharply in location to location.

This paper humbly attempts to make a very accurate on location, carbon monitoring service, in free of charge for people who interested to monitor such as Researchers, Naturists, Students, Decision makers etc., who tries to control carbon emission to save the mother earth. However, the development of this system will be critically depends on resources which are available. We make more innovative solutions to sustain and overcome those constrains.

5.7 Further Developments

This system can easily improve as a one stop solution for Air Quality monitoring system which can monitor multiple gases with particular matters. The inner program architecture is developed well to plug any kind of sensor with least changes to its processor. A Wireless device can be added to connect this system with pc, freely with an optional solar panel. An internal rechargeable battery pack can be installed to enhance mobility. Finally, a suitable design of industrial cover/case will qualify this for the market. ,

5.8 References

[1] http://www.modbus.org/specs.php (Accessed on January 7, 2012)

[2] http://aprs.gids.nl/nmea/ (Accessed on January 7, 2012)

[3] Scott Meyers, "Effective C++ second Edition"Addison Wesley Longman, Inc. Harlow, England,1998.

[4] Richered Barnett, Sarah Cox and Larry O'Cull, "Embedded C Programming and the Atmel AVR", Delmer Cengage Learning, NewYork,USA,2007.

[5] "GSU-52 Series" White paper module, Position Co.Ltd., Tokyo , Japan,2006.

[6] "LM35 Precision Centigrade Temperature Sensors" Data sheet, National Semiconductor Corporation, USA, and November 2000.

[7] "EM_CO2-Engine_K30-STA" Data sheet, Sense Air AB Co,Delsbo,Sweden,

[8] "SUNLIKE SC1602BSLB" Data sheet,2000.

[9] http://zfacts.com/p/226.html (Accessed on Junary 14, 2012)

[10] Chan et al., 2002 C.W. Chan, Y. Peng, L.L. Chen Knowledge acquisition and ontology modeling for construction of a control and monitoring expert system International Journal of Systems Science, 33 (2002), pp. 485–503

[11] (FAO, 2010a;Hett, Castella, Heinimann, Messerli, & Pfund,2012;Le Quere, Raupach, Canadell, & Marland, 2009;Van derWerf et al., 2009).

[12] Sholeh et al., 2007M. Sholeh, H.F. Svendsen, A.H. Karl, J. OlavSelection of new absorbents for carbon dioxide capture ,Energy Conversion and Management, 48 (2007), pp. 251–258

[13] Zhou et al., 2009Q. Zhou, C.W. Chan, P. Tontiwachi wuthikul A monitoring and diagnostic expert system for carbon dioxide capture Expert System with Applications, 36 (2009), pp. 1621–1631

[14] FAO, 2010a;Hett, Castella, Heinimann, Messerli, & Pfund,2012;Le Quere, Raupach, Canadell, & Marland, 2009;Van derWerf et al., 2009).

[15] IPCC (Intergovernmental Panel on Climate Change)IPCC special report on carbon dioxide capture and storageCambridge University Press, New York (2005) Available at: http://www.ipcc.ch/pdf/special-reports/srccs/srccs_wholereport.pdf. (Accessed on July 4, 2013)

[16] NETL (National Energy Technology Laboratory)Best practices for monitoring, verification, and accounting of CO_2 stored in deep geologic formations National Energy Technology Laboratory, Pittsburgh (PA) (2009) Available at: http://www.netl.doe.gov/technologies/carbon_seq/refshelf/MVA_Document.pdf. (Accessed on June 7, 2013)

[17] D.W. Vasco, A. Rucci, A. Ferretti, F. Novali, R.C. Bissel, P.S. Ringrose et al.Satellite-based measurements of surface deformation reveal fluid flow associated with the geological storage of carbon dioxide Geophys Res Lett, 37 (2010), p. L03303 http://dx.doi.org/10.1029/2009GL041544 , (Accessed on August 29, 2013)

[18] M.D. Morris Factorial sampling plans for preliminary computational experiments Technometrics, 33 (2) (1991), pp. 161–174 http://dx.doi.org/10.2307/1269043 (Accessed on August 29, 2013)

[19] Arduino. (n.d.). Retrieved from http://www.arduino.cc/.(Accessed on August 29, 2013)

[20] Schmidt, M. "Arduino: A Quick Start Guide", Pragmatic Bookshelf, January 22 2011, Pg. 201.

[21] Arduinowifishield.(2013).Retrieved from http://arduino.cc/en/Main/ArduinoWiFiShield. (Accessed on, sep 9, 2013)

[22] Sainsmart mega2560 board 3.2 tft lcd module display shield kit for atmel atmega avr 16au atmega8u2. (2013). Retrieved from http://henningkarlsen.com/electronics/library.php?id=52. (Accessed on Sep 9, 2013)

[23] WiFi.ScanNetworks. (N.D). Retrieved from http://arduino.cc/de/Reference/WiFiScanNetworks. (Accessed on Sep 12, 2013)

[24] Wifi.ssid. (N.D). Retrieved from http://arduino.cc/en/Reference/WiFiSSID.

[25] Connectwith wpa. (N.D). Retrieved from http://arduino.cc/en/Tutorial/ScanNetworks. (Accessed on Sep 14, 2013)

[27] O. Gruber ASDEX Upgrade enhancements in view of ITER application Fusion Engineering and Design, 84 (2009)

[28] V. Bobkov, ICRF operation with improved antennas in a full W-wall ASDEX Upgrade – status and developments, in: 24th IAEA FEC, San Diego, in press.

[29] Arduino, Homepage: http://arduino.cc. (Accessed on Sep 24, 2013)

[30] J.P. Aikio, T. Rahkonen A comprehensive analysis of AM–AM and AM–PM conversion in an LDMOS RF power amplifier IEEE Transactions on Microwave Theory and Techniques, 57 (February (2)) (2009)

[31] D. Grine Improvement of the phase regulation between two amplifiers feeding the inputs of the 3 dB combiner in the ASDEX-Upgrade ICRH system AIP Conference Proceedings, vol. 1406 (2011), p. 105 View Record in Scopus | Cited By in Scopus (1)

[32] K. Polozhiy Influence of the phase shift between antennas on W sputtering in ASDEX Upgrade 38th EPS Conference on Plasma Physics (2011)

[33] http://arduino.cc/en/Main/ArduinoBoardDue (Accessed on sep 29, 2013)

[34] W. Mccarter, O. Vennesland Sensor systems for use in reinforced concrete structures Constr Build Mater, 18 (6) (2004), pp. 351–358

[35] Quinn B, Kelly G. Feasibility of embedded wireless sensors for monitoring of concrete curing and structural health. In: Sensors and smart structures technologies for civil, mechanical, and aerospace systems, San Diego, USA, March 2010.

[36] Datasheet Filter Cap SF2 for Humidity and Temperature Sensor SHT2x, December 2011.
<http://www.sensirioncom/en/pdf/product_information/Datasheet_filter-cap_sf2.pdf >. (Accessed on sep 29, 2013)

Bibliography

The Program Code

6 The Main Program

```
#include <stdlib.h>
#include <arduino.h>
#include "co2.h"
#include "lcd.h"
#include "at24c.h"

#define DEVICE_ID "0000000000000000"

/*************************GLOBAL*****************************/
LCD lcd;
AT24C at24c;
int32_t eeprom_next_address=-1;
uint8_t mpx_a0; //mpx address
uint8_t mpx_a1; //mpx address
#define GSM_POWERKEY 8
#define SERVER "www.sliit.net"
/************************************************************/

int readGPS(unsigned long interval,char* GPSdata)
{
  //if(!(GPSdata=(char*)malloc(512)))return -2;
  unsigned short index=0;
  boolean captured=false;
  boolean ready_to_capture=false;
  boolean capture_enable=false;
  unsigned long timeout=800;
  Serial.begin(9600);
  //Serial.println("Reading GPS...");

  unsigned long lastCharTime=millis();

  while(true)
  {
    if((millis()-lastCharTime)>timeout)
    {
      ready_to_capture=false;
      capture_enable=false;
```

```
  //Serial.println("Timeout");
  return -1;
}
//Serial.println("interval = "+String(millis()-lastCharTime));
if((millis()-lastCharTime)>interval)
{
  if(captured)
  {
    Serial.flush();
    Serial.end();
    break;
    //return 0;
    //sREG=SREG;
    //cli();
  }
  else
  {
    if(ready_to_capture)
    {
      capture_enable=true;
      //lastCharTime=millis();
    }
  }
}

////////////////////////////////////
if(!capture_enable)
{
  //Serial.println("¥n!capture_enable");
  while(Serial.available())
  {
    Serial.read();
    ready_to_capture=true;
    //Serial.write(Serial.read());
    lastCharTime=millis();
  }
}
else
{
  //Serial.println("¥ncapture_enable");
  while(Serial.available())
  {
    GPSdata[index]=Serial.read();
    index++;
    //Serial.write(Serial.read());
    ready_to_capture=true;
```

```
          lastCharTime=millis();
          captured=true;
        }
      }
      ////////////////////////////////////

    }
    GPSdata[index]='¥0';
    //prints(GPSdata);
    return 0;
  }

  int getRecordCount(char* sourceChars)
  {
    int i=0;
    int recordCount=0;
    while(sourceChars[i]!='¥0')
    {
      if(sourceChars[i]==',')recordCount++;
      i++;
    }
    i=0;
    if(recordCount>0)recordCount++;
    return recordCount;
  }

  GPRMC getGPRMC(char* GPSdata);
  GPRMC getGPRMC(char* GPSdata)
  {
    GPRMC gprmc;

    char GPSline[512];

if(subString(GPSdata,indexOf(GPSdata,"$GPRMC"),indexOf(GPSdata,"¥r¥n"),GPSli
ne)<0)
      if(subString(GPSdata,indexOf(GPSdata,"$GPRMC"),GPSline)<0)
      {
        printsln("No GPRMC sentence.");
        delay(500);
        //return (struct GPRMC_t){0};
        return gprmc;
      }
    //printsln(GPSline);
    int recordCount;
    if((recordCount=getRecordCount(GPSline))!=13)
    {
```

```
        printsln("No 13 records. Found ");
        prints(recordCount);
        prints(".");
        //return (struct GPRMC_t){0};
        return gprmc;
      }

      char time[7];

subString(GPSline,nthIndexOf(GPSline,',',1)+1,nthIndexOf(GPSline,',',1)+7,ti
me);
      char time2[]={

time[0],time[1],':',time[2],time[3],':',time[4],time[5],'¥0'             };
      copyString(time2,gprmc.time);
      //printsln("¥nTime: "); prints(gprmc.time);delay(1000);

subString(GPSline,nthIndexOf(GPSline,',',2)+1,nthIndexOf(GPSline,',',3),gprm
c.validity);
      //prints(gprmc.validity);

      char latitude[10];

subString(GPSline,nthIndexOf(GPSline,',',3)+1,nthIndexOf(GPSline,',',4),lati
tude);
      char lat[3];
      subString(latitude,0,2,lat);
      char latMins[8];
      subString(latitude,2,latMins);
      gprmc.latitude=atof(lat)+(atof(latMins)/60);
      char north[2];

subString(GPSline,nthIndexOf(GPSline,',',4)+1,nthIndexOf(GPSline,',',5),nort
h);
      if(north[0]=='S')gprmc.latitude*=-1;
      //printsln("Latitude:¥n");prints(gprmc.latitude);delay(1000);

      char longitude[11];

subString(GPSline,nthIndexOf(GPSline,',',5)+1,nthIndexOf(GPSline,',',6),long
itude);
      char longi[4];
      subString(longitude,0,3,longi);
      char longiMins[8];
      subString(longitude,3,longiMins);
      gprmc.longitude=atof(longi)+(atof(longiMins)/60);
```

```
      char east[2];

subString(GPSline,nthIndexOf(GPSline,',',6)+1,nthIndexOf(GPSline,',',7),east
);
      if(east[0]=='W')gprmc.longitude*=-1;
      //printsln("Longitude:¥n");prints(gprmc.longitude);delay(1000);

      char speed_in_knots[6];

subString(GPSline,nthIndexOf(GPSline,',',7)+1,nthIndexOf(GPSline,',',8),spee
d_in_knots);
      gprmc.speed_in_kmph=atof(speed_in_knots)*1.852;
      //printsln("Speed (km/h):¥n");prints(gprmc.speed_in_kmph);delay(1000);

      char course_made_good_true[6];

subString(GPSline,nthIndexOf(GPSline,',',8)+1,nthIndexOf(GPSline,',',9),cour
se_made_good_true);
      gprmc.course_made_good_true=atof(course_made_good_true);
      //prints(gprmc.course_made_good_true);

      char date[7];

subString(GPSline,nthIndexOf(GPSline,',',9)+1,nthIndexOf(GPSline,',',10),dat
e);
      char date2[]={

date[0],date[1],'/',date[2],date[3],'/',date[4],date[5],'¥0'             };
      copyString(date2,gprmc.date);
      //printsln("Date: ");prints(gprmc.date);
      //  printsln("Time: "); prints(gprmc.time);delay(1000);

      char magnetic_variation[6];

if(subString(GPSline,nthIndexOf(GPSline,',',10)+1,nthIndexOf(GPSline,',',11)
,magnetic_variation)<=0)magnetic_variation[0]='¥0';
      gprmc.magnetic_variation=atof(magnetic_variation);
      //prints(gprmc.magnetic_variation);


subString(GPSline,nthIndexOf(GPSline,'*',1)+1,nthIndexOf(GPSline,'*',1)+3,gp
rmc.checksum);
      //prints(gprmc.checksum);

      return gprmc;
    }
```

```
    GPGGA getGPGGA(char* GPSdata);
    GPGGA getGPGGA(char* GPSdata)
    {
      GPGGA gpgga;

      char GPSline[512];

if(subString(GPSdata,indexOf(GPSdata,"$GPGGA"),indexOf(GPSdata,'*',indexOf(G
PSdata,"$GPGGA"))+3,GPSline)<0)
        //if(subString(GPSdata,indexOf(GPSdata,"$GPGGA"),GPSline)<0)
      {
        printsln("No GPGGA sentence.");
        delay(500);
        //return (struct GPRMC_t){0};
        return gpgga;
      }
      //printsln(GPSline);
      int recordCount;
      if((recordCount=getRecordCount(GPSline))!=15)
      {
        printsln("GPGGA not 15 records. Found ");
        prints(recordCount);
        prints(".");
        //return (struct GPRMC_t){0};
        return gpgga;
      }

      char time[7];

subString(GPSline,nthIndexOf(GPSline,',',1)+1,nthIndexOf(GPSline,',',1)+7,ti
me);
      char time2[]={

time[0],time[1],':',time[2],time[3],':',time[4],time[5],'¥0'              };
      copyString(time2,gpgga.time);
      //printsln("¥nTime: "); prints(GPRMC.time);delay(1000);

      char latitude[10];

subString(GPSline,nthIndexOf(GPSline,',',2)+1,nthIndexOf(GPSline,',',3),lati
tude);
      char lat[3];
      subString(latitude,0,2,lat);
      char latMins[8];
      subString(latitude,2,latMins);
```

```
      gpgga.latitude=atof(lat)+(atof(latMins)/60);
      char north[2];

subString(GPSline,nthIndexOf(GPSline,',',3)+1,nthIndexOf(GPSline,',',4),nort
h);
      if(north[0]=='S')gpgga.latitude*=-1;
      //printsln("Latitude:¥n");prints(gpgga.latitude);delay(1000);

      char longitude[11];

subString(GPSline,nthIndexOf(GPSline,',',4)+1,nthIndexOf(GPSline,',',5),long
itude);
      char longi[4];
      subString(longitude,0,3,longi);
      char longiMins[8];
      subString(longitude,3,longiMins);
      gpgga.longitude=atof(longi)+(atof(longiMins)/60);
      char east[2];

subString(GPSline,nthIndexOf(GPSline,',',5)+1,nthIndexOf(GPSline,',',6),east
);
      if(east[0]=='W')gpgga.longitude*=-1;
      //printsln("Longitude:¥n");prints(gpgga.longitude);delay(1000);

      char fix_quality[2];

subString(GPSline,nthIndexOf(GPSline,',',6)+1,nthIndexOf(GPSline,',',7),fix_
quality);
      gpgga.fix_quality=atoi(fix_quality);
      //printsln("Fix quality:¥n");prints(gpgga.fix_quality);delay(1000);

      char number_of_satellites[3];

subString(GPSline,nthIndexOf(GPSline,',',7)+1,nthIndexOf(GPSline,',',8),numb
er_of_satellites);
      gpgga.number_of_satellites=atoi(number_of_satellites);
      //prints(gpgga.number_of_satellites);

      char hdop[6];

subString(GPSline,nthIndexOf(GPSline,',',8)+1,nthIndexOf(GPSline,',',9),hdop
);
      gpgga.hdop=atof(hdop);
      //prints(gpgga.hdop);

      char altitude[8];
```

```
subString(GPSline,nthIndexOf(GPSline,',',9)+1,nthIndexOf(GPSline,',',10),alt
itude);
      gpgga.altitude=atof(altitude);
      //prints(gpgga.altitude);

      char geoid_height[8];

subString(GPSline,nthIndexOf(GPSline,',',11)+1,nthIndexOf(GPSline,',',12),ge
oid_height);
      gpgga.geoid_height=atof(geoid_height);
      //prints(gpgga.geoid_height);

      char time_since_last_dgps_update[7];

subString(GPSline,nthIndexOf(GPSline,',',13)+1,nthIndexOf(GPSline,',',14),ti
me_since_last_dgps_update);
      gpgga.time_since_last_dgps_update=atoi(time_since_last_dgps_update);
      //prints(gpgga.time_since_last_dgps_update);


subString(GPSline,nthIndexOf(GPSline,',',14)+1,nthIndexOf(GPSline,',',14)+5,
gpgga.dgps_reference_station_id);
      //prints(gpgga.dgps_reference_station_id);


subString(GPSline,nthIndexOf(GPSline,'*',1)+1,nthIndexOf(GPSline,'*',1)+3,gp
gga.checksum);
      //printsln(gpgga.checksum);

      return gpgga;
    }

    GPS getGPS();
    GPS getGPS()
    {

      switchMultiplexer(0);//Switch multiplexer to GPS

      GPS gps;
      GPRMC gprmc;
      GPGGA gpgga;
      char* GPSdata=(char*)malloc(512);
      //char* GPSline;
      if(readGPS(100,GPSdata)<0){
        printsln("Can't read from GPS...");
```

```
        delay(1000);
        printsln(" ");
      }
      else
      {
        //printsln(GPSdata);
        gprmc=getGPRMC(GPSdata);
        //printsln("GPRMC Checksum: ");prints(gprmc.checksum);delay(1000);
        /*
        printsln("GPRMC Latitude:¥n");prints(gprmc.latitude);delay(1000);
         printsln("GPRMC Longitude:¥n");prints(gprmc.longitude);delay(1000);
         printsln("GPRMC Speed
(km/h):¥n");prints(gprmc.speed_in_kmph);delay(1000);
         printsln("GPRMC Date: ");prints(gprmc.date);
         printsln("GPRMC Time: "); prints(gprmc.time);delay(1000);
         */

        gps.latitude=gprmc.latitude;
        gps.longitude=gprmc.longitude;
        gps.speed_in_kmph=gprmc.speed_in_kmph;
        copyString(gprmc.date,gps.date);
        copyString(gprmc.time,gps.time);

        gpgga=getGPGGA(GPSdata);
        //printsln("GPGGA Checksum: ");prints(gpgga.checksum);delay(1000);

        //printsln("GPGGA Altitude:¥n");prints(gpgga.altitude);delay(1000);

        gps.altitude=gpgga.altitude;
        if((gprmc.checksum[0]!='¥0')&&(gpgga.checksum[0]!='¥0'))
        {
          if(gpgga.fix_quality>0&&gpgga.number_of_satellites>2)
          {
            gps.dirty=false;
          }
        }
      }
      free(GPSdata);
      switchMultiplexer(3); //switch to unused port
      return gps;
    }

    int getAnalogCO2()
    {
      return 2000.0*analogRead(A0)/850.7276507276507;
    }
```

```
int getCO2()
{
  switchMultiplexer(1);//Switch multiplexer to CO2

  int CO2;
  Serial.begin(9600,SERIAL_8N2);
  Serial.write(0xfe);
  Serial.write(0x04);
  Serial.write(0x00);
  Serial.write(0x03);
  Serial.write(0x00);
  Serial.write(0x01);
  Serial.write(0xd5);
  Serial.write(0xc5);
  Serial.flush();

  for(int i=0;Serial.available()<5;i++)
  {
    delay(60);
    if(i>2){
      Serial.end();
      return -1;
    }
  }
  Serial.read();
  Serial.read();
  Serial.read();
  CO2=Serial.read();
  CO2=(CO2<<8)+Serial.read();
  Serial.end();
  switchMultiplexer(3); //switch to unused port
  return CO2;
}

int getTemperature()
{
  return 150.0*analogRead(A7)/319.022869022869; //for 4.815V supply
  //use 307.2 for 5.000V supply
}

void _gsm_power_pulse()
{
  pinMode(GSM_POWERKEY,OUTPUT);
  digitalWrite(GSM_POWERKEY,HIGH);
  delay(500);
```

```
  digitalWrite(GSM_POWERKEY,LOW);
  delay(1000);
  digitalWrite(GSM_POWERKEY,HIGH);
}

int _gsm(char* command, char* expectedResponse, unsigned long timeout)
{
  uint16_t SIZE=256;
  char readbuffer[SIZE];
  uint16_t index;
  unsigned long starttime;

  index=0;
  starttime=millis();

  if(command!="")
  {
    while(Serial.available())Serial.read();
    Serial.println(command);
    Serial.flush();
  }

  while(millis()-starttime<timeout)
  {
    while(Serial.available())
    {
      if(index<SIZE)
      {
        readbuffer[index]=Serial.read();
        index++;
        if(readbuffer[index-1]=='¥n')
        {
          readbuffer[index]='¥0';
          index=0;
          if(strstr(readbuffer,expectedResponse)!=NULL)
          {
            while(Serial.available())Serial.read();
            return 0;
          }
        }
      }
    }
    readbuffer[index]='¥0';
    if(strstr(readbuffer,expectedResponse)!=NULL)
    {
      while(Serial.available())Serial.read();
```

```
      return 0;
    }
  }
  return -1;
}

int _gsm_power_on()
{
  boolean success=false;
  unsigned long starttime;
  uint16_t net_login_timeout=20000;
  for(uint8_t i=0;i<2;i++)
  {
    _gsm_power_pulse();
    starttime=millis();
    while(millis()-starttime<net_login_timeout)
      if(_gsm("AT+CREG?","+CREG: 0,1",1000)==0)
      {
        success=true;
        blink13(32);
        break;
      }

    if(success)break;
  }
  return success?0:-1;
}

int gsm_boot_or_reboot()
{
  boolean success=false;
  Serial.begin(9600);
  Serial.flush();
  switchMultiplexer(2);
  while(Serial.available())Serial.read();

  if(_gsm("AT","OK",2000)==0)
  {
    _gsm_power_pulse();
    _gsm("","NORMAL POWER DOWN",5000);
    delay(1000);
  }

  success=_gsm_power_on()==0;

  Serial.end();
```

```
  switchMultiplexer(3);
  if(success)return 0;
  return -1;
}

int gsm_power_on()
{
  Serial.begin(9600);
  Serial.flush();
  switchMultiplexer(2);
  while(Serial.available())Serial.read();

  int gsm_power_status=0;
  if(_gsm("AT+CREG?","+CREG: 0,1",1000)!=0)
  {
    gsm_power_status=_gsm_power_on();
  }
  Serial.end();
  switchMultiplexer(3);

  printsln("GSM: ");
  if(gsm_power_status==0)prints("READY");
  else prints(gsm_power_status);
  delay(2000);

  return gsm_power_status;
}

int gsm_power_down()
{
  boolean success=false;
  Serial.begin(9600);
  Serial.flush();
  switchMultiplexer(2);
  while(Serial.available())Serial.read();

  if(_gsm("AT","OK",2000)==0)
  {
    _gsm_power_pulse();
    _gsm("","NORMAL POWER DOWN",5000);
    delay(1000);
    success=true;
  }
  Serial.end();
  switchMultiplexer(3);
  if(success)return 0;
```

```
  return -1;
}

void blink13(uint8_t times)
{
  pinMode(13,OUTPUT);
  for(uint8_t i=0;i<times;i++)
  {
    digitalWrite(13,HIGH);
    delay(25);
    digitalWrite(13,LOW);
    delay(50);
  }
}

int upload(EEPROM_RECORD er)
{
  printsln("Uploading...");

  unsigned char* buffer=(unsigned char*)&er;
  int length=sizeof(EEPROM_RECORD);
  char eeprom_record_hex[length*2+1];
  for(int i=0;i<length;i++)
  {
    sprintf(&eeprom_record_hex[i*2],"%02X",buffer[i]);
  }

  eeprom_record_hex[length*2]=0;
  char get_request[192];
  strcpy(get_request,"GET /airboy/upload.php?packet=");
  strcat(get_request,eeprom_record_hex);
  int checksum=0;
  for(int i=0;i<length*2;i++)
  {
    checksum+=(int)eeprom_record_hex[i];
  }
  unsigned char* checksumPointer=(unsigned char*)&checksum;
  char checksumString[5];
  for(int i=0;i<2;i++)
  {
    sprintf(&checksumString[i*2],"%02X",checksumPointer[i]);
  }
  checksumString[4]=0;
  strcat(get_request,checksumString);
  strcat(get_request,DEVICE_ID);
  strcat(get_request," HTTP/1.1¥r¥nHOST: ");
```

```
      strcat(get_request,SERVER);
      strcat(get_request,"¥r¥n¥r¥n¥x1A");

      Serial.begin(9600);
      Serial.flush();

      switchMultiplexer(2); //switch multiplexer to GPRS
      boolean success=false;
      _gsm("¥x1A","",2000);
      _gsm("AT+CIPCLOSE","CLOSE OK",2000);
      char cipstartline[128];
      strcpy(cipstartline,"AT+CIPSTART=¥"TCP¥",¥"");
      strcat(cipstartline,SERVER);
      strcat(cipstartline,"¥",¥"80¥"");
      if(_gsm(cipstartline,"CONNECT OK",5000)==0)
        if(_gsm("AT+CIPSEND",">",5000)==0)
          if(_gsm(get_request,"HTTP/1.1 200 OK",10000)==0)
            success=true;
      Serial.end();
      switchMultiplexer(3); //switch to unused port

      if(success)printsln("HTTP/1.1 200 OK");
      //else printsln("Upload failed.");
      if(success)return 0;
      return -1;
    }

    int8_t eeprom_format()
    {
      eeprom_next_address=0;
      int8_t i = at24c.write_byte(0,0x03);
      printsln("Format complete. Waiting 10sec.");
      delay(10000);
      return i;
    }

    int8_t eeprom_save(EEPROM_RECORD er)
    {
      if(at24c.write_byte(eeprom_next_address,0x1e)<0)return -1;
      eeprom_next_address++;
      char buffer[sizeof(EEPROM_RECORD)];
      memcpy(buffer,&er,sizeof(EEPROM_RECORD));

if(at24c.write_bytes(eeprom_next_address,buffer,sizeof(EEPROM_RECORD))<0)ret
urn -1;
      eeprom_next_address+=sizeof(EEPROM_RECORD);
```

```
  if(at24c.write_byte(eeprom_next_address,0x03)<0)return -1;
  return 0;
}

EEPROM_RECORD eeprom_open(uint32_t record_id)
{
  uint32_t record_address=record_id*(sizeof(EEPROM_RECORD)+1)+1;
  return eeprom_open_address(record_address);
}

EEPROM_RECORD eeprom_open_last()
{
  uint32_t record_address=eeprom_next_address-sizeof(EEPROM_RECORD);
  return eeprom_open_address(record_address);
}

EEPROM_RECORD eeprom_open_address(uint32_t record_address)
{
  char buffer[sizeof(EEPROM_RECORD)];
  at24c.read_bytes(record_address,buffer,sizeof(EEPROM_RECORD));
  EEPROM_RECORD er;
  memcpy(&er,buffer,sizeof(EEPROM_RECORD));
  return er;
}

/*
void eeprom_dump_json()
{
  if(eeprom_next_address==0)
  {
    printsln("No records in   EEPROM to dump.");
    delay(5000);
    printsln(" ");
    return;
  }
  uint32_t record_address;
  uint32_t last_address=eeprom_next_address-sizeof(EEPROM_RECORD);

  printsln((int)(last_address/sizeof(EEPROM_RECORD)));
  prints(" records will be dumped.");
  delay(2000);

  prints("Memory dump in..",0);
  char countdown[3];
  for(int i=5;i>0;i--)
  {
```

```
  sprintf(countdown,"%d",i);
  prints(countdown,1);delay(1000);
}
delay(3000);

Serial.begin(9600);
Serial.flush();

Serial.println("[");
for(uint32_t record_id=0;true;record_id++)
{
  EEPROM_RECORD er;
  record_address=record_id*(sizeof(EEPROM_RECORD)+1)+1;
  if(record_address<=last_address)
    er=eeprom_open_address(record_address);
  else
  {
    Serial.println("]");
    return;
  }
  Serial.println(" {");

  Serial.print("  Latitude: ");
  Serial.print(er.latitude);Serial.println(",");
  Serial.print("  Longitude: ");
  Serial.print(er.longitude);Serial.println(",");
  Serial.print("  Altitude: ");
  Serial.print(er.altitude);
  Serial.print("m");Serial.println(",");
  Serial.print("  Speed: ");
  Serial.print(er.speed_in_kmph);
  Serial.print("Km/h");Serial.println(",");
  Serial.print("  Date: ");
  Serial.print(er.date);Serial.println(",");
  Serial.print("  Time: ");
  Serial.print(er.time);Serial.println(",");
  Serial.print("  Temperature: ");
  Serial.print(er.temperature);
  Serial.print("Â°C");Serial.println(",");
  Serial.print("  CO2: ");
  Serial.print(er.CO2);
  Serial.println("ppm");

  Serial.println(" }");
  Serial.println(",");
}
```

```
  Serial.flush();
  Serial.end();
}*/
/*****************************************************************************
*************************/
```

```
void setup()
{
  mpx_a0=9;//mpx
  mpx_a1=10;//mpx
  lcd.init(7,6,5,4,3,2);
  at24c.init(11,12,131072,256);

  printsln("RESET");
  delay(500);

  //eeprom_format();
  printsln("Searching next  EEPROM address..");
  eeprom_next_address=at24c.search_byte(0x03);
  int state;
  if(eeprom_next_address==-1)
  {
    printsln("Formatting...");
    state=eeprom_format();
    if(state<0)
    {
      printsln("EEPROM ERROR: ");
      prints(state);
    }
  }
  delay(5000);
}

void loop()
{
  printsln("Reading GPS...  ");
  GPS gps=getGPS();
  if(!gps.dirty)
  {
    gsm_power_on();
```

```
EEPROM_RECORD er;
er.latitude=gps.latitude;
er.longitude=gps.longitude;
er.altitude=gps.altitude;
er.speed_in_kmph=gps.speed_in_kmph;
copyString(gps.date,er.date);
copyString(gps.time,er.time);
er.temperature=getTemperature();
er.CO2=getCO2();//getAnalogCO2();

//eeprom_save(er);

/*
printsln("Saving..");
if(eeprom_save(er)<0)
{
  printsln("EEPROM write failure.");
}
else
{
  printsln("EEPROM write success.");
}
//delay(5000);

er=eeprom_open_last();
*/

printsln("Latitude:¥n");
prints(er.latitude);
delay(1000);
printsln("Longitude:¥n");
prints(er.longitude);
delay(1000);
printsln("Altitude:¥n");
prints(er.altitude);
prints("m");
delay(1000);
printsln("Speed:¥n");
prints(er.speed_in_kmph);
prints("Km/h");
delay(1000);
printsln("Date: ");
prints(er.date);
printsln("Time: ");
prints(er.time);
delay(1000);
```

```
        printsln("Temperature:¥n");
        prints(er.temperature);
        prints("'C");
        delay(1000);
        printsln("CO2: ");
        prints(er.CO2);
        prints("ppm");
        delay(1000);
        //printsln("A. CO2:
");prints(getAnalogCO2());prints("ppm");delay(1000);

        if(isValid(er))
        {
          if(upload(er)<0)
          {
            printsln("Upload failed.");
            printsln("GSM Rebooting...");
            gsm_boot_or_reboot();
          }
        }else{
          printsln("Invalid data.");
        }
        delay(5000);
        delayMinutesWhileShowingCO2(1);
      }
      else
      {
        printsln("Invalid GPS data");
        if(gsm_power_down()==0)printsln("GSM shutdown...");
        //if(getCO2()<0)eeprom_dump_json();
        //else
          _printCO2andTemperature();
      }
    }

    //helper functions...
    boolean isValid(EEPROM_RECORD er)
    {
      if(er.CO2>0)return true;

      return false;
    }

    void delayMinutesWhileShowingCO2(unsigned long minutes)
    {
      char sleep_text[17];
```

```
  char temp_text[3];

  for(unsigned long i=0;i<minutes;i++)
  {
    for(uint8_t j=0;j<(60);j++)
    {
      _printCO2andTemperature();

      strcpy(sleep_text,"Sleeping ");
      if(j==0)sprintf(temp_text,"%d",(int)(minutes-i));
      else sprintf(temp_text,"%d",(int)(minutes-1-i));
      strcat(sleep_text,temp_text);
      strcat(sleep_text,":");
      if((60-j)%60<10)strcat(sleep_text,"0");
      sprintf(temp_text,"%d",((60-j)%60));
      strcat(sleep_text,temp_text);
      prints(sleep_text,1);

      delay(1000);
    }
  }
}

void _printCO2andTemperature()
{
  char CO2_and_temperature[17];
  sprintf(CO2_and_temperature,"%dppm %d'C",getCO2(),getTemperature());
  lcd.clear();
  prints(CO2_and_temperature,0);
}

void switchMultiplexer(uint8_t address){
  pinMode(mpx_a0,OUTPUT);
  pinMode(mpx_a1,OUTPUT);
  digitalWrite(mpx_a0,address & 0b0000001 ? HIGH : LOW);
  digitalWrite(mpx_a1,address & 0b0000010 ? HIGH : LOW);
}

void prints(char* text,uint8_t line)
{
  Serial.begin(9600);
  Serial.println(text);
  Serial.flush();
  Serial.end();

  char print_text[17];
```

```
  sprintf(print_text,"%-16s",text);
  lcd.cursor(0,line);
  lcd.print(print_text);
}

void prints(char* text)
{

  Serial.begin(9600);
  Serial.print(text);
  Serial.flush();
  Serial.end();

  lcd.print(text);
}
void printsln(char* text)
{

  Serial.begin(9600);
  Serial.print("¥r¥n");
  Serial.print(text);
  Serial.flush();
  Serial.end();

  lcd.println(text);
}

void prints(int value)
{
  char text[10];
  itoa(value,text,sizeof(text));
  prints(text);
}
void printsln(int value)
{
  char text[10];
  itoa(value,text,sizeof(text));
  printsln(text);
}

void prints(double value)
{
  char text[13];
  dtostrf(value,12,7,text);
  prints(text);
}
```

```
    void printsln(double value)
    {
      char text[13];
      dtostrf(value,12,7,text);
      printsln(text);
    }

    int subString(char sourceString[], int startIndex,int endIndex,char*
destinationString)
    {
      if(startIndex>=0&&startIndex<endIndex)
      {
        int i;
        for(i=startIndex;i<endIndex;i++)
        {
          destinationString[i-startIndex]=sourceString[i];
        }
        destinationString[i-startIndex]='¥0';
        return i-startIndex;
      }
      return -1;
    }

    int subString(char sourceString[], int startIndex,char* destinationString)
    {
      if(startIndex>=0)
      {
        int i;
        for(i=startIndex;i<strlen(sourceString);i++)
        {
          destinationString[i-startIndex]=sourceString[i];
        }
        destinationString[i-startIndex]='¥0';
        return i-startIndex;
      }
      return -1;
    }

    int copyString(char sourceString[], char* destinationString)
    {
      return subString(sourceString,0,destinationString);
    }

    int indexOf(char sourceString[],char searchString[],int startIndex)
    {
      int searchStringLength=strlen(searchString);
```

```
      int sourceStringLength=strlen(sourceString);
      if((searchStringLength<sourceStringLength)&&searchStringLength>0)
      {
        int MAXLENGTH=4096;
        int _maxlength=MAXLENGTH-1;
        int index=startIndex;
        int i;

while(index+searchStringLength<sourceStringLength&&index+searchStringLength!
=_maxlength)
        {
          for(i=0;i<searchStringLength;i++)
          {
            if(sourceString[i+index]!=searchString[i])break;
          }
          if(i==searchStringLength)return index;
          index++;
        }
      }
      return -1;
    }

    int indexOf(char sourceString[],char searchString[])
    {
      return indexOf(sourceString,searchString,0);
    }

    int indexOf(char sourceString[],char searchChar,int startIndex)
    {
      int MAXLENGTH=4096;
      int _maxlength=MAXLENGTH-1;
      int index=startIndex;
      while(sourceString[index]!='¥0'&&index!=_maxlength)
      {
        if(sourceString[index]==searchChar)return index;
        index++;
      }
      return -1;
    }

    int indexOf(char sourceString[],char searchChar)
    {
      return indexOf(sourceString,searchChar,0);
    }

    int nthIndexOf(char sourceString[],char searchChar,int n,int startIndex)
```

```
    {
      //prints(n);prints(startIndex);
      if(n<=0||(indexOf(sourceString,searchChar,startIndex)<0))return -1;
      if(n==1)return indexOf(sourceString,searchChar,startIndex);
      return
nthIndexOf(sourceString,searchChar,n-1,indexOf(sourceString,searchChar,start
Index)+1);
    }
    int nthIndexOf(char sourceString[],char searchChar,int n)
    {
      return nthIndexOf(sourceString,searchChar,n,0);
    }

    int _free2DCharArray(char** arr,int count)
    {
      for(int i=0;i<count;i++)
      {
        free(arr[i]);
      }
      free(arr);
    }
    int availableMemory() {
      int mem = 2048; // Use 2048 with ATmega328
      byte *buf;

      while ((buf = (byte *) malloc(--mem)) == NULL)
        ;
      free(buf);

      return mem;
    }

    int serialMonitor()
    {
      Serial.begin(9600);
      Serial.println("Serial Monitor...");
      while(true)
      {
        while(Serial.available())
        {
          Serial.write(Serial.read());
        }
      }
      Serial.end();}
```

6.2 at24c.h

```
#include "i2c.h"

/*
  Ex: for AT24C1024B EEPROM
  at24c.init(11,12,131072,256);
*/

class AT24C
{
  public:
  int init(uint8_t scl_pin,uint8_t sda_pin,uint32_t memory_size,uint16_t
page_size)
  {
    i2c.init(scl_pin,sda_pin);
    SIZE=memory_size;
    PAGE_SIZE=page_size;
  }

  uint8_t read_byte(uint32_t address)
  {
    if(address>SIZE-1)return '¥0';

    uint8_t bits;

    i2c.start_condition();
    i2c.write_byte(0b10100000|((address>>15)&0b00001110));
    i2c.write_byte((address>>8)&0xff);
    i2c.write_byte(address & 0xff);
    i2c.start_condition();
    i2c.write_byte(0b10100001|((address>>15)&0b00001110));
    bits=i2c.read_byte();
    i2c.no_ack();
    i2c.stop_condition();

    return bits;
  }

  char* read_bytes(uint32_t address, char buffer[], uint32_t length)
  {
    if((address+length)>SIZE-1){buffer[0]='¥0';return buffer;}

    i2c.start_condition();
```

```
  i2c.write_byte(0b10100000|((address>>15)&0b00001110));
  i2c.write_byte((address>>8)&0xff);
  i2c.write_byte(address & 0xff);
  i2c.start_condition();
  i2c.write_byte(0b10100001|((address>>15)&0b00001110));
  for(int i=0;i<length;i++)
  {
    buffer[i]=i2c.read_byte();
    (i<(length-1))?i2c.ack():i2c.no_ack();
  }
  i2c.stop_condition();
  return buffer;
}

int32_t search_byte(uint8_t bits)
{
  return search_byte(bits,0,1);
}

int32_t search_byte(uint8_t bits,uint32_t start_address)
{
  return search_byte(bits,start_address,1);
}

int32_t search_byte(uint8_t bits,uint32_t start_address,uint32_t nth)
{
  if(start_address>SIZE-1)return -2;
  if(!(nth>=0))return -3;

  uint32_t n=0;
  i2c.start_condition();
  i2c.write_byte(0b10100000|((start_address>>15)&0b00001110));
  i2c.write_byte((start_address>>8)&0xff);
  i2c.write_byte(start_address & 0xff);
  i2c.start_condition();
  i2c.write_byte(0b10100001|((start_address>>15)&0b00001110));
  for(uint32_t i=start_address;i<SIZE;i++)
  {
    if(bits==i2c.read_byte())
    {
      n++;
      if(n!=nth)
      {
        (i<SIZE-1)?i2c.ack():i2c.no_ack();
        continue;
      }
```

```
      i2c.no_ack();
      i2c.stop_condition();
      return i;
    }
    (i<SIZE-1)?i2c.ack():i2c.no_ack();
  }
  i2c.stop_condition();
  return -1;
}

int8_t write_byte(uint32_t address,uint8_t bits)
{
  if(address>SIZE-1)return -1;

  i2c.start_condition();
  i2c.write_byte(0b10100000|((address>>15)&0b00001110));
  i2c.write_byte((address>>8)&0xff);
  i2c.write_byte(address & 0xff);
  i2c.write_byte(bits);
  i2c.stop_condition();
  delayMicroseconds(5000);

  if(read_byte(address)==bits)return 0;
  return -2; //EEPROM error
}

int8_t write_bytes(uint32_t address,char buffer[], uint32_t length)
{
  if((address+length)>SIZE-1)return -1;

  boolean firstrun=true;
  i2c.start_condition();
  i2c.write_byte(0b10100000|((address>>15)&0b00001110));
  i2c.write_byte((address>>8)&0xff);
  i2c.write_byte(address & 0xff);
  for(uint32_t offset=0;offset<length;offset++)
  {
    if(firstrun)firstrun=false;
    else
    {
      if(!((address+offset)&(PAGE_SIZE-1)))
      {
        i2c.stop_condition();
        delayMicroseconds(5000);
        i2c.start_condition();
        i2c.write_byte(0b10100000|(((address+offset)>>15)&0b00001110));
```

```
            i2c.write_byte(((address+offset)>>8)&0xff);
            i2c.write_byte((address+offset) & 0xff);
          }
        }
        i2c.write_byte(buffer[offset]);
      }
      i2c.stop_condition();
      delayMicroseconds(5000);

      return 0;
    }

    private:
    I2C i2c;
    uint32_t SIZE;
    uint16_t PAGE_SIZE;
};
```

6.3. co2.h

```
#include <arduino.h>

typedef struct GPRMC
{
  char time[9];
  char validity[2];
  double latitude;
  double longitude;
  int speed_in_kmph;
  double course_made_good_true;
  char date[9];
  double magnetic_variation;
  char checksum[3];
  GPRMC()
  {
    time[0]='¥0';
    validity[0]='¥0';
    latitude=0.0;
    longitude=0.0;
    speed_in_kmph=0.0;
    course_made_good_true=0.0;
    date[0]='¥0';
    magnetic_variation=0.0;
    checksum[0]='¥0';
  }
};

typedef struct GPGGA
{
  char time[9];
  double latitude;
  double longitude;
  unsigned short fix_quality;
  unsigned short number_of_satellites;
  double hdop;
  int altitude;
  double geoid_height;
  int time_since_last_dgps_update;
  char dgps_reference_station_id[5];
  char checksum[3];
  GPGGA()
  {
    time[0]='¥0';
```

```
    latitude=0.0;
    longitude=0.0;
    fix_quality=0;
    number_of_satellites=0;
    hdop=0.0;
    altitude=0.0;
    geoid_height=0.0;
    time_since_last_dgps_update=0;
    dgps_reference_station_id[0]='¥0';
    checksum[0]='¥0';
  }
};

typedef struct GPS
{
  char date[9];
  char time[9];
  double latitude;
  double longitude;
  int altitude;
  int speed_in_kmph;
  boolean dirty;
  GPS(){dirty=true;}
};

typedef struct EEPROM_RECORD
{
  char date[9];
  char time[9];
  double latitude;
  double longitude;
  int16_t altitude;//int16_t
  uint8_t speed_in_kmph;//uint8_t
  int8_t temperature;//int8_t
  int16_t CO2;//int16_t
};
```

6.4 I2c.h

```
#include <arduino.h>

class I2C
{
  public:
  int init(uint8_t scl_pin,uint8_t sda_pin)
  {
    pinMode(sda_pin,INPUT);
    pinMode(scl_pin,OUTPUT);
    digitalWrite(scl_pin,LOW);
    digitalWrite(sda_pin,HIGH);
    SCL=scl_pin;
    SDA=sda_pin;
    return 0;
  }

  void start_condition()
  {
    //Serial.println("start_condition()");
    digitalWrite(SCL,LOW);
    pinMode(SDA,OUTPUT);
    digitalWrite(SDA,HIGH);
    //interval();
    digitalWrite(SCL,HIGH);
    //interval();
    digitalWrite(SDA,LOW);
    //interval();
    digitalWrite(SCL,LOW);
    //interval();
    pinMode(SDA,INPUT);
    digitalWrite(SDA,HIGH);
  }

  void stop_condition()
  {
    //Serial.println("stop_condition()");
    digitalWrite(SCL,LOW);
    pinMode(SDA,OUTPUT);
    digitalWrite(SDA,LOW);
    //interval();
    digitalWrite(SCL,HIGH);
    //interval();
    digitalWrite(SDA,HIGH);
```

```
  //interval();//2
  digitalWrite(SCL,LOW);
  //interval();
  pinMode(SDA,INPUT);
  digitalWrite(SDA,HIGH);
}

void ack()
{
  //Serial.println(" ack()");
  digitalWrite(SCL,LOW);
  pinMode(SDA,OUTPUT);
  digitalWrite(SDA,LOW);
  //interval();
  digitalWrite(SCL,HIGH);
  //interval();
  digitalWrite(SCL,LOW);
  //interval();
  pinMode(SDA,INPUT);
  digitalWrite(SDA,HIGH);
}

void no_ack()
{
  //Serial.println(" no_ack()");
  digitalWrite(SCL,LOW);
  pinMode(SDA,OUTPUT);
  digitalWrite(SDA,HIGH);
  //interval();
  digitalWrite(SCL,HIGH);
  //interval();
  digitalWrite(SCL,LOW);
  //interval();
  pinMode(SDA,INPUT);
  digitalWrite(SDA,HIGH);
}

void reset()
{
  Serial.println("I2C Protocol Reset.");
  digitalWrite(SCL,LOW);
  pinMode(SDA,OUTPUT);
  digitalWrite(SDA,HIGH);

  //interval();
  digitalWrite(SCL,HIGH);
```

```
  //interval();
  digitalWrite(SDA,LOW);
  //interval();
  digitalWrite(SCL,LOW);
  //interval();
  digitalWrite(SDA,HIGH);
  //interval();

  for(int i=0;i<9;i++)
  {
  digitalWrite(SCL,HIGH);
  //interval();
  digitalWrite(SCL,LOW);
  //interval();
  }

  digitalWrite(SCL,HIGH);
  //interval();
  digitalWrite(SDA,LOW);
  //interval();
  digitalWrite(SCL,LOW);
  //interval();
  digitalWrite(SCL,HIGH);
  //interval();
  digitalWrite(SDA,HIGH);
  //interval();
  digitalWrite(SCL,LOW);
  //interval();

  pinMode(SDA,INPUT);
  digitalWrite(SDA,HIGH);
}

uint8_t read_byte()
{
  pinMode(SDA,INPUT);
  digitalWrite(SDA,HIGH);
  //while(digitalRead(SDA)!=LOW);
  char temp[9];
  //Serial.println("read_byte()------------");
  uint8_t bits=0x00;
  //interval();
  for(int8_t i=7;i>=0;i--)
  {
    digitalWrite(SCL,HIGH);
    //interval();
```

```
          //Serial.print("read_byte()");Serial.println(digitalRead(SDA));
          bits|=(digitalRead(SDA) << i);
          //Serial.println(digitalRead(SDA));

//Serial.print("read_byte()");Serial.println(digitalRead(SDA)==HIGH);
          //Serial.println(itoa(bits,temp,2));
          digitalWrite(SCL,LOW);
          //interval();
        }

        //no_ack();
        //Serial.println(itoa(bits,temp,2));
        return bits;
      }

      boolean write_byte(uint8_t bits)
      {
        //char temp[9];
        //Serial.println("write_byte()-------------");
        //Serial.println(itoa(bits,temp,2));

        //pinMode(SDA,INPUT);
        //digitalWrite(SDA,HIGH);
        //while(digitalRead(SDA)!=LOW);

        digitalWrite(SCL,LOW);
        digitalWrite(SDA,LOW);
        pinMode(SDA,OUTPUT);
        //interval();
        for(uint8_t i=0;i<8;i++)
        {
          digitalWrite(SDA,(bits<<i) & 0x80);
          //Serial.print("write_byte()");Serial.println(((bits<<i) &
0x80)==0x80));
          //interval();
          digitalWrite(SCL,HIGH);
          //interval();
          digitalWrite(SCL,LOW);
          //interval();
        }

        pinMode(SDA,INPUT);
        digitalWrite(SDA,HIGH);
        //interval();
        digitalWrite(SCL,HIGH);
```

```
    //interval();
    //Serial.print("ack: ");Serial.println(digitalRead(SDA));
    while(digitalRead(SDA)!=LOW);
    //Serial.print("ack: ");Serial.println(digitalRead(SDA));



    digitalWrite(SCL,LOW);
    //interval();

    return digitalRead(SDA)==LOW;
  }

  private:
  uint8_t SCL;
  uint8_t SDA;


};
```

6.5 lcd.h

```
7  #include <arduino.h>
8
9  class LCD
10 {
11   public:
12     int init(uint8_t rs, uint8_t en, uint8_t d7, uint8_t d6, uint8_t d5, uint8_t
   d4)
13     {
14       pinMode(rs,OUTPUT);
15       pinMode(en,OUTPUT);
16       pinMode(d7,OUTPUT);
17       pinMode(d6,OUTPUT);
18       pinMode(d5,OUTPUT);
19       pinMode(d4,OUTPUT);
20       digitalWrite(rs,LOW);
21       digitalWrite(en,LOW);
22       digitalWrite(d7,LOW);
23       digitalWrite(d6,LOW);
24       digitalWrite(d5,LOW);
25       digitalWrite(d4,LOW);
26       RS=rs;
27       EN=en;
28       D7=d7;
29       D6=d6;
30       D5=d5;
31       D4=d4;
32
33       x=0;
34       y=0;
35
36       for(uint8_t i=0;i<16;i++)old_row_1[i]=' ';
37
38       init_sequence();
39       return 0;
40     }
41
42     void print(char character)
43     {
44       if(character=='¥r')cursor(0,y);
```

```
45      else if(character=='¥n')cursor(0,++y);
46      else data_write(character);
47    }
48
49    void println(char character)
50    {
51      print('¥n');
52      print(character);
53    }
54
55    void print(char* text)
56    {
57      int length=strlen(text);
58      for(int i=0; i<length; i++)
59      {
60        if(text[i]=='¥r')cursor(0,y);
61        else if(text[i]=='¥n')cursor(0,++y);
62        else data_write(text[i]);
63      }
64    }
65
66    void println(char* text)
67    {
68      print('¥n');
69      print(text);
70    }
71
72    void print(int value)
73    {
74      char text[7];
75      itoa(value,text,10);
76      print(text);
77    }
78
79    void println(int value)
80    {
81      print('¥n');
82      print(value);
83    }
84
```

```
85    void print(long value)
86    {
87      char text[12];
88      ltoa(value,text,10);
89      print(text);
90    }
91
92    void println(long value)
93    {
94      print('¥n');
95      print(value);
96    }
97
98    void cursor(uint8_t x,uint8_t y)
99    {
100     /*
101     if(x>15)x=0;
102     if(y>1)y=0;
103     */
104     LCD::x=x;
105     LCD::y=y;
106     instruction_write(LCD_SETDDRAMADDR | y << 6 | x);
107   }
108
109   void cursor_on()
110   {
111     instruction_write(LCD_DISPLAYCONTROL | LCD_CURSORON);
112   }
113
114   void cursor_off()
115   {
116     instruction_write(LCD_DISPLAYCONTROL | LCD_CURSOROFF);
117   }
118
119   void clear()
120   {
121     instruction_write(LCD_CLEARDISPLAY);
122   }
123
124 private:
```

```
125   uint8_t RS;
126   uint8_t EN;
127   uint8_t D7;
128   uint8_t D6;
129   uint8_t D5;
130   uint8_t D4;
131
132   uint8_t x;
133   uint8_t y;
134
135   uint8_t old_row_1[16];
136
137   // commands
138   static const byte LCD_CLEARDISPLAY = 0x01;
139   static const byte LCD_RETURNHOME = 0x02;
140   static const byte LCD_ENTRYMODESET = 0x04;
141   static const byte LCD_DISPLAYCONTROL = 0x08;
142   static const byte LCD_CURSORSHIFT = 0x10;
143   static const byte LCD_FUNCTIONSET = 0x20;
144   static const byte LCD_SETCGRAMADDR = 0x40;
145   static const byte LCD_SETDDRAMADDR = 0x80;
146
147   // flags for display entry mode
148   static const byte LCD_ENTRYRIGHT = 0x00;
149   static const byte LCD_ENTRYLEFT = 0x02;
150   static const byte LCD_ENTRYSHIFTINCREMENT = 0x01;
151   static const byte LCD_ENTRYSHIFTDECREMENT = 0x00;
152
153   // flags for display on/off control
154   static const byte LCD_DISPLAYON = 0x04;
155   static const byte LCD_DISPLAYOFF = 0x00;
156   static const byte LCD_CURSORON = 0x02;
157   static const byte LCD_CURSOROFF = 0x00;
158   static const byte LCD_BLINKON = 0x01;
159   static const byte LCD_BLINKOFF = 0x00;
160
161   // flags for display/cursor shift
162   static const byte LCD_DISPLAYMOVE = 0x08;
163   static const byte LCD_CURSORMOVE = 0x00;
164   static const byte LCD_MOVERIGHT = 0x04;
```

```
165   static const byte LCD_MOVELEFT = 0x00;
166
167   // flags for function set
168   static const byte LCD_8BITMODE = 0x10;
169   static const byte LCD_4BITMODE = 0x00;
170   static const byte LCD_2LINE = 0x08;
171   static const byte LCD_1LINE = 0x00;
172   static const byte LCD_5x10DOTS = 0x04;
173   static const byte LCD_5x7DOTS = 0x00;
174
175   void init_sequence()
176   {
177     delay(100);
178     digitalWrite(RS,LOW);
179     digitalWrite(D7,LOW);
180     digitalWrite(D6,LOW);
181     digitalWrite(D5,HIGH);
182     digitalWrite(D4,HIGH);
183     enable_pulse();
184     delayMicroseconds(5000); //>4.1ms
185     enable_pulse();
186     delayMicroseconds(150); //>100us
187     enable_pulse();
188     delayMicroseconds(150); //>100us
189
190     digitalWrite(D4,LOW);
191     enable_pulse();
192     delayMicroseconds(150); //>100us
193
194     instruction_write(LCD_FUNCTIONSET | LCD_4BITMODE | LCD_2LINE |
   LCD_5x7DOTS);
195     instruction_write(LCD_DISPLAYCONTROL | LCD_DISPLAYOFF | LCD_CURSOROFF
   | LCD_BLINKOFF);
196     instruction_write(LCD_CLEARDISPLAY);
197     instruction_write(LCD_ENTRYMODESET | LCD_ENTRYLEFT |
   LCD_ENTRYSHIFTDECREMENT);
198     //init ends here.
199
200     instruction_write(LCD_DISPLAYCONTROL | LCD_DISPLAYON | LCD_CURSOROFF |
   LCD_BLINKOFF);
```

```
201   }
202
203   void instruction_write(uint8_t instruction)
204   {
205     digitalWrite(RS,LOW);
206     write(instruction);
207     if((instruction & LCD_CLEARDISPLAY) || (instruction & LCD_RETURNHOME))
   delayMicroseconds(5000);
208   }
209
210   void data_write(uint8_t data)
211   {
212     if(x>15)
213     {
214       y++;x=0;
215       cursor(x,y);
216     }
217     if(y>1)
218     {
219       clear();
220       digitalWrite(RS,HIGH);
221       for(uint8_t i=0;i<16;i++)
222       {
223         write(old_row_1[i]);
224         old_row_1[i]=' ';
225       }
226       y=1;
227       cursor(x,y);
228     }
229
230
231     if(y==1)old_row_1[x]=data;
232     digitalWrite(RS,HIGH);
233     write(data);
234     x++;
235   }
236
237   void write(uint8_t value)
238   {
239     digitalWrite(D7,(value >> 7) & 0x01);
```

```
240     digitalWrite(D6,(value >> 6) & 0x01);
241     digitalWrite(D5,(value >> 5) & 0x01);
242     digitalWrite(D4,(value >> 4) & 0x01);
243     enable_pulse();
244     digitalWrite(D7,(value >> 3) & 0x01);
245     digitalWrite(D6,(value >> 2) & 0x01);
246     digitalWrite(D5,(value >> 1) & 0x01);
247     digitalWrite(D4,(value >> 0) & 0x01);
248     enable_pulse();
249     delayMicroseconds(100);
250   }
251
252   void enable_pulse()
253   {
254     digitalWrite(EN,HIGH);
255     delayMicroseconds(100);
256     digitalWrite(EN,LOW);
257     delayMicroseconds(100);
258   }
259};
260
261/*
262//Example implementaion..
263LCD lcd;
264void setup()
265{
266  lcd.init(12,11,5,4,3,2); //(rs,en,d7,d6,d5,d4)
267}
```

11 . Web Interface Programming

11.1. index.php

```
<!DOCTYPE html>
<?php
```

```
        include "./dbcon/dbcon.php";
        $dbcon=new dbcon();

        $result=$dbcon->get_data("SELECT * FROM telemetry_data ORDER BY timestamp
DESC                                                                    LIMIT
".(isset($_GET["page"])&&$_GET["page"]>1?(($_GET["page"]-1)*100).",":"")."10
0");

        function add_br($value)
        {
          return $value.'<br />';
        }
    ?>
    <html>
    <style>
    #telemetry
    {
    font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
    width:100%;
    border-collapse:collapse;
    }
    #telemetry td, #telemetry th
    {
    font-size:1em;
    border:1px solid #98bf21;
    padding:3px 7px 2px 7px;
    }
    #telemetry th
    {
    font-size:1.1em;
    text-align:left;
    padding-top:5px;
    padding-bottom:4px;
    background-color:#A7C942;
    color:#ffffff;
    }
    #telemetry tr td
    {
    background-color:#204020;
    }
    #telemetry tr.alt td
    {
    background-color:#408040;
    }
    sub
    {
```

```
    font-size:0.6em;
    }
    .button
    {
      background-color: #30a030;
      -moz-border-radius: 3px;
      -webkit-border-radius: 3px;
      border-radius:4px;
      color: #00ff00;
      font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
      font-size: 10px;
      text-decoration: none;
      cursor: poiner;
      border: none;
    }
    </style>
    <script type="text/JavaScript" language="JavaScript">
    openLinkGotCalled=false;
    function openLink(url)
    {
      openLinkGotCalled=true;
      //var win=window.open(url, '_blank');
      //win.focus();
      document.getElementById("mapview").src=url;
      document.getElementById("mapview").style.display="block";
    }
    </script>
    <body style="background-color: #000000; color: #00ff00">
    <div                 style="position:absolute;margin:                 50px
50px;left:0;right:0;top:0;bottom:0;overflow:auto;"
onclick="if(!openLinkGotCalled){document.getElementById('mapview').style.dis
play='none';document.getElementById('mapview').src='img/brilliant_color_lens
_03_vector_161589.jpg';}else openLinkGotCalled=false;">

    <div  style="font-family:'Trebuchet  MS',  Arial,  Helvetica,  sans-serif;
font-size: 1.5em; text-align: center;">GPS Assisted Online CO<sub>2</sub> /
Temperature Mapping System</div><br />
    <span     style="font-family:'Trebuchet     MS',     Arial,     Helvetica,
sans-serif;">AIRBOY CO<sub>2</sub> TELEMETRY DATA</span><br /><br />
    <?php
       if ($result) {
         echo                                                          "<table
id='telemetry'>¥n<tr><th>UTC</th><th>Latitude</th><th>Longitude</th><th>Alti
tude                (m)</th><th>Speed                (km/h)</th><th>Temperature
(C&deg;)</th><th>CO<sub>2</sub> (ppm)</th><th>See on map</th></tr>¥n";
```

```
          $styleAlt=true;
          while($row = mysql_fetch_array($result))
          {
             echo $styleAlt?"<tr class='alt'>¥n":"<tr>¥n";
             $styleAlt=!$styleAlt;
   /*
             $count = count($row);
             for($i=0;$i<$count/2;$i++)
             {
               $field=$row[$i];
               echo "   <td>$field</td>¥n";
             }
   */
             $datetime=$row["timestamp"];        $latitude=$row["latitude"];
$longitude=$row["longitude"];                    $altitude=$row["altitude"];
$speed_in_kmph=$row["speed_in_kmph"];      $temperature=$row["temperature"];
$CO2=$row["CO2"];
             ?>
             <td><?php echo $datetime; ?></td>
             <td><?php printf('%.6F',$latitude); ?></td>
             <td><?php printf('%.6F',$longitude); ?></td>
             <td><?php printf('%.1F',$altitude); ?></td>
             <td><?php echo $speed_in_kmph; ?></td>
             <td><?php echo $temperature; ?></td>
             <td><?php echo $CO2; ?></td>
             <td><input     type="button"     class="button"     value="MAP"
onclick="openLink('https://maps.google.com/?ie=UTF8&amp;ll=<?php
printf('%.6F',$latitude); ?>,<?php printf('%.6F',$longitude); ?>&amp;q=<?php
printf('%.6F',$latitude);                                          ?>,<?php
printf('%.6F',$longitude);  ?>&amp;spn=4.833408,10.821533&amp;t=m&amp;z=14&am
p;output=embed');" /></td>
             <?php
             //foreach($row as $field)echo "   <td>$field</td>¥n";
             //print_r(array_map('add_br',$row));
             echo "</tr>¥n";
          }
        echo "</table>";

        ?>
             <div style="text-align: right;">
                <a              href="index.php?page=<?php              echo
isset($_GET["page"])&&$_GET["page"]>1?$_GET["page"]-1:'1'; ?>">[ &lt; ]</a>
                <a              href="index.php?page=<?php              echo
isset($_GET["page"])?$_GET["page"]+1:'2'; ?>">[ &gt; ]</a>
             </div>
        <?php
```

```
    }
?>

<br />
<span style="font-family:'Trebuchet MS', Arial, Helvetica, sans-serif; font-size: 0.7em;">
The objective of this research is to develop a free global access method for internet users to have accurate in-situ CO2 density measurements in certain points of the world, to force relevant parties to reduce Carbon Dioxide emission to save the mother earth from global warming.<br />
This system has been developed as a part of KISSEL project, Kanssei Mathematics laboratory, Ibaraki University, Japan. Visit the test<a href="analyze.php">data analysis page</a> to see how this data canbe used.
</span>

</div>


<iframe style="display: none; width: 425px; height: 350px; margin: 0 auto; left: 0; right: 0; top: 110px; position: absolute; z-index: 100; box-shadow: 0px 0px 10px #000000;" id="mapview" width="425" height="350" frameborder="0" scrolling="no" marginheight="0" marginwidth="0" src="img/brilliant_color_lens_03_vector_161589.jpg"></iframe>

</body></html>
```

11.2 upload.php

```
<?php

function hex2bin($hex)
{
  $arr=str_split($hex);
  $result="";
  $count=count($arr);
  for($i=0;$i<($count/2);$i++)
  {
    $result.=chr(base_convert($arr[$i*2].$arr[$i*2+1],16,10));
  }
  return $result;
}
```

```
    function hexTo32Float($strHex) {
       $v = hexdec($strHex);
       $x = ($v & ((1 << 23) - 1)) + (1 << 23) * ($v >> 31 | 1);
       $exp = ($v >> 23 & 0xFF) - 127;
       return $x * pow(2, $exp - 23);
    }

    /*
      char date[9];
      char time[9];
      double latitude;
      double longitude;
      int16_t altitude;//int16_t
      uint8_t speed_in_kmph;//uint8_t
      int8_t temperature;//int8_t
      int16_t CO2;//int16_t
    */

       $packet              =              $_GET['packet'];              //ex:
31382f30332f37370031383a31363a303000a0a0a0a0b0b0b0b0c0c0d0e0f0f0
       $dateHexString = substr($packet,0,18);
       $timeHexString = substr($packet, 18,18);
       $dateString = hex2bin($dateHexString);
       $timeString = hex2bin($timeHexString);
       $dateString = str_split($dateString);
       $datetime                                                              =
'20'.$dateString[6].$dateString[7].'-'.$dateString[3].$dateString[4].'-'.$da
teString[0].$dateString[1].' '.$timeString.'+00:00';

       //echo $datetime;

       $latitude = implode(unpack("f",pack('H*',substr($packet, 36,8))));
       $longitude = implode(unpack("f",pack('H*',substr($packet, 44,8))));
       $altitude   =   hexdec(substr($packet,   52,2))+hexdec(substr($packet,
54,2))*16*16;
       $speed_in_kmph = hexdec(substr($packet, 56,2));
       $temperature = hexdec(substr($packet, 58,2));
       $CO2     =     hexdec(substr($packet,     60,2))+hexdec(substr($packet,
62,2))*16*16;
       $checksum   =   hexdec(substr($packet,   64,2))+hexdec(substr($packet,
66,2))*16*16;
       $deviceiid = substr($packet, 68,16);

       include "./dbcon/dbcon.php";
       $dbcon=new dbcon();
```

```
	$result=$dbcon->set_data("INSERT INTO telemetry_data
VALUES('$datetime',$latitude,$longitude,$altitude,$speed_in_kmph,$temperatur
e,$CO2)");

	$result=$dbcon->get_data("SELECT * FROM telemetry_data");

	function add_br($value)
	{
	  return $value.'<br />';
	}

	//<html><body><span style="font-family: monospace; font-size: 8pt;">
	/*if ($result) {
	   while($row = mysql_fetch_array($result))
	   {
	      print_r(array_map('add_br',$row));
	      echo "<br />";
	   }
	}*/

	echo "You said: ".$_GET['packet']."¥r¥n¥r¥n";
	echo "datetime: $datetime¥r¥n";
	echo "latitude: $latitude¥r¥n";
	echo "longitude: $longitude¥r¥n";
	echo "altitude: $altitude¥r¥n";
	echo "speed_in_kmph: $speed_in_kmph¥r¥n";
	echo "temperature: $temperature¥r¥n";
	echo "CO2: $CO2¥r¥n";
	echo "checksum: $checksum¥r¥n";
	echo "device_id: $device_id¥r¥n";



	//</span></body></html>
?>
```