Seeeduino Stalker

Overview

Seeeduino Stalker is an extensive Wireless Sensor Network node, with native data logging features and considered modular structure. It could be conveniently used as sensor hub or integrated data collector for monitoring, tracking and etc.

Basic features:

- Arduino compatible, based on Seeeduino
- Native RTC, On board CR2032 battery holder
- Native micro SD card socket
- Easy stackable I2C pin header
- Dual voltage (5V/3.3V) I2C hub
- Standalone and shield mode
- User defined button and LED
- B" series socket (2*10 pin 2.0mm pitch, same as Xbee)
- Interrupt evoking ready
- Serial interface with DTR for auto reset during programming

License

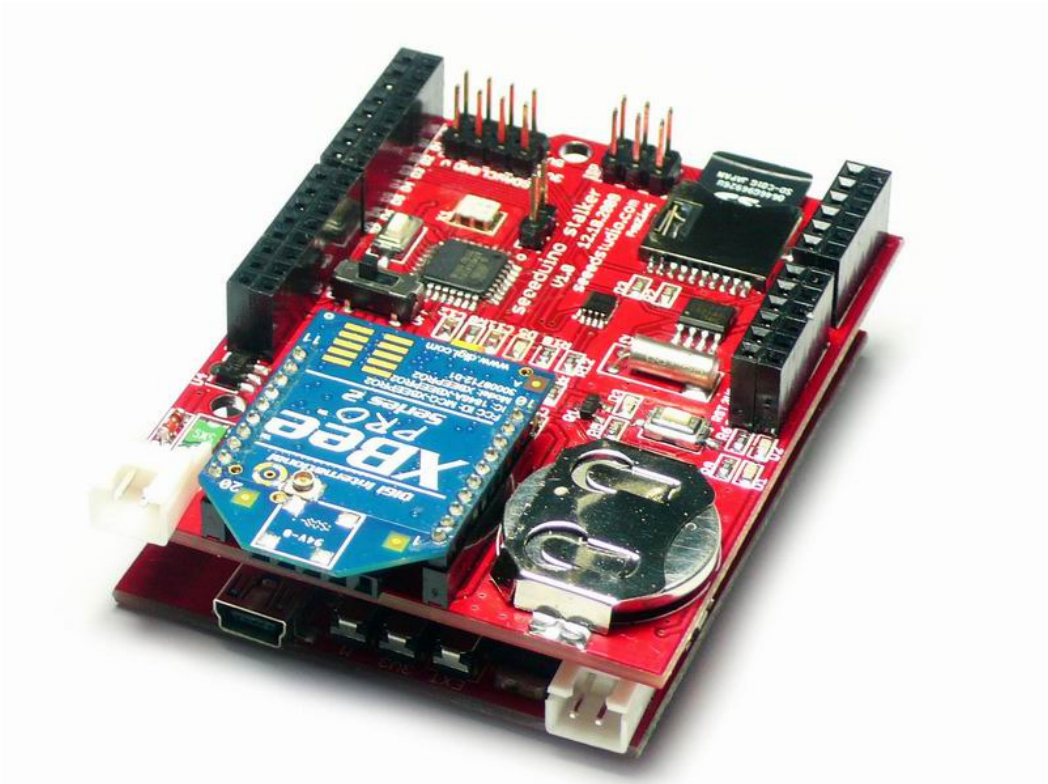
Source files and documents are licensed under a Creative Commons Attribution 3.0 Unported License.

Specifications

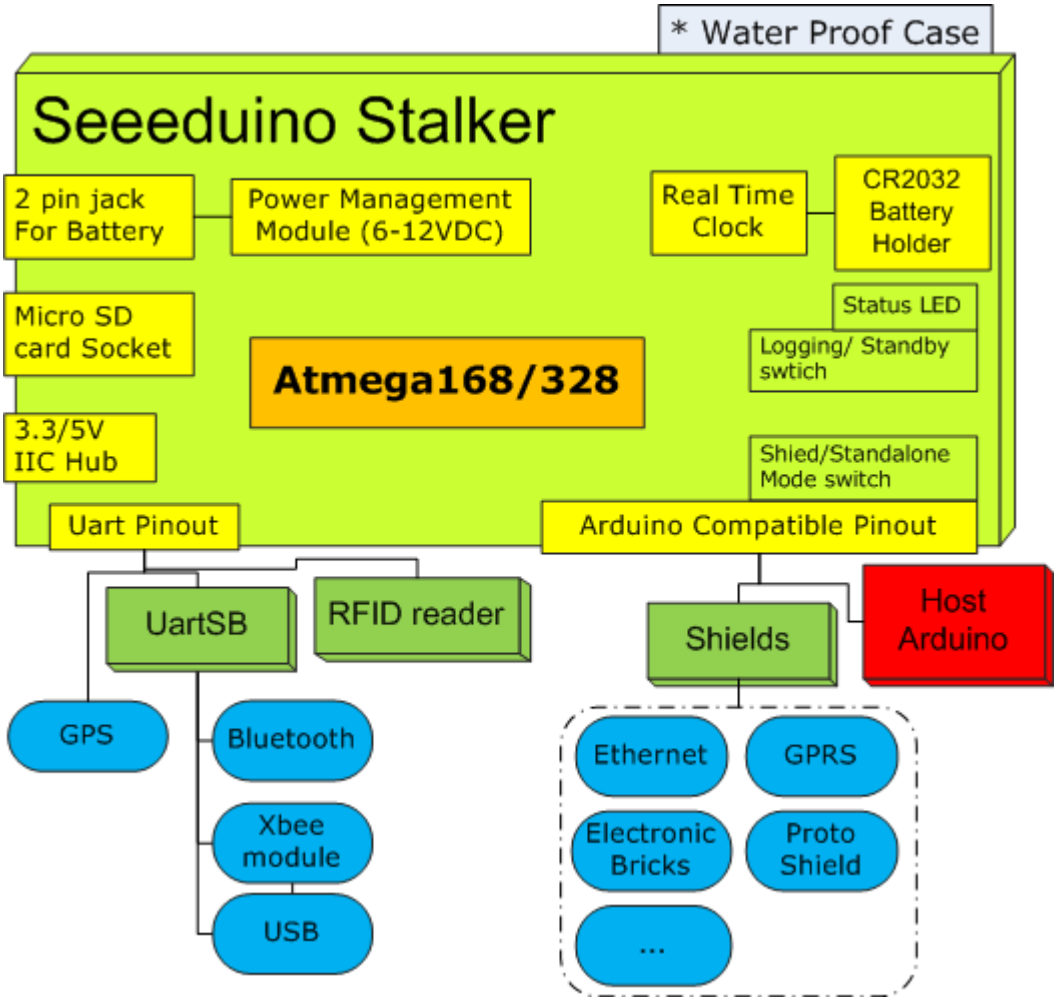
Microprocessor	ATMega 168
PCB size	6.8cm x 5.5cm x 0.16cm
Indicators	Reset, Power, Pin 13LED
Power supply	5V or 7-12V
Power Connector	2 pin JST/ USB
IO counts	20
ADC input	Dedicated 2 channel 10bit resolution
Connectivity	I2C, Uart, SPI
Communication Protocol	I2C, Uart
RHOS	Yes

Electrical Characterstics

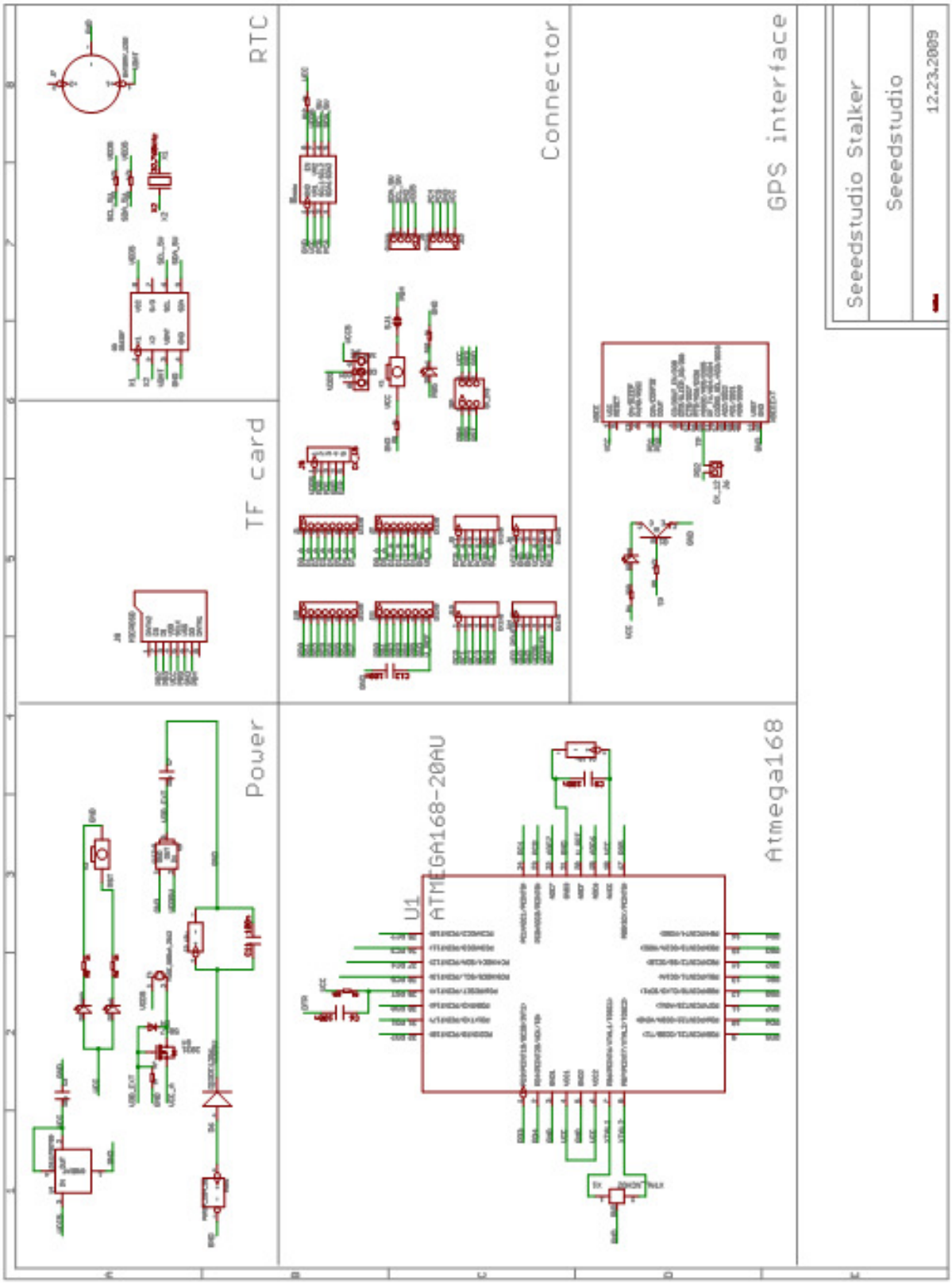
Specification	Min	Typ	Max	Unit
Input voltage	5	9	12	VDC
Global Current Consumption		300	1000	mA
3.3V IIC voltage	3.2	3.3	3.5	VDC
5V IIC voltage	4.6	4.7	5	VDC
Uart Baud Rate			115200	bps



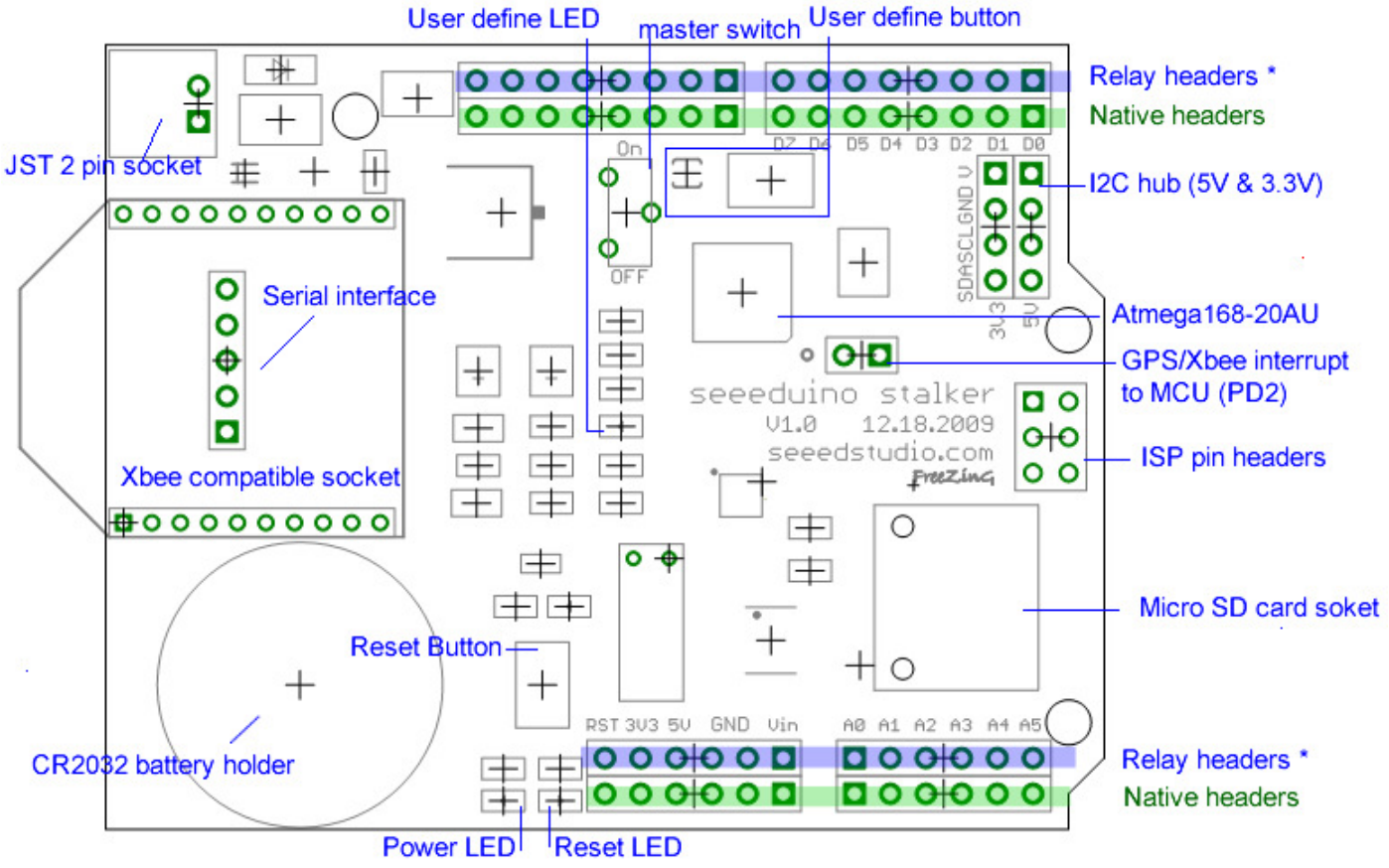
System Block Diagram



Scheme


RTC
TF card
Power
Connector
GPS interface
Atmega168
U1
ATMEGA168-20AU
Seeedstudio Stalker
Seeedstudio
12.23.2009

Hardware Installation



*Relay headers are transparent pin holes except I2C are connected, Native headers are full pin-out of Stalker MCU

XBEE compatible socket – 2*10 2.0mm pitch female header allows add capability of XBEE modules and many compatible one like GPS B", RF B", Blue B" and many etc. They usually communicate with Stalker via serial port.

Serial interface – USB connectivity is not provided by default, to save space and also cost. However you could use UartSB or other USB to serial adapter to do the programming or link with PC.

User defined LED and button – User defined LED and button are prepared to minimize the efforts to make a standalone device.

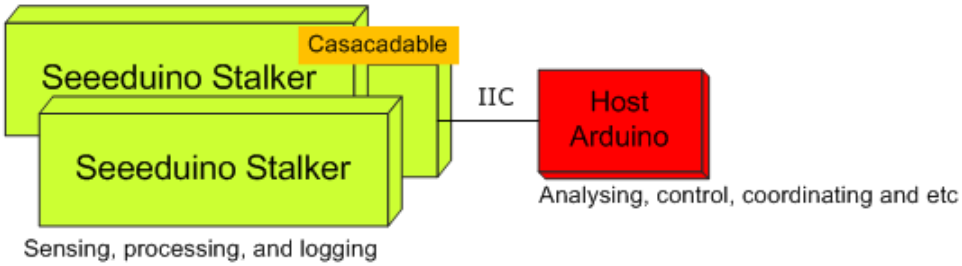
Master switch – Turns off stalker, RTC will remain working if battery inserted.

GPS/Xbee interrupt to MCU – Default open, connect when MCU needs to response on B" modules by interrupt.

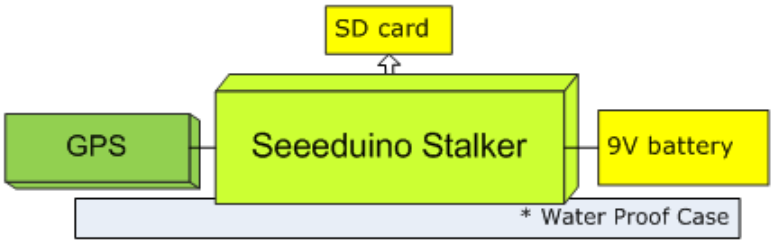
I2C hub (5V/3.3V): Onboard I2C level shifter IC provides seamless bus between 3.3V and 5V devices.

Applications

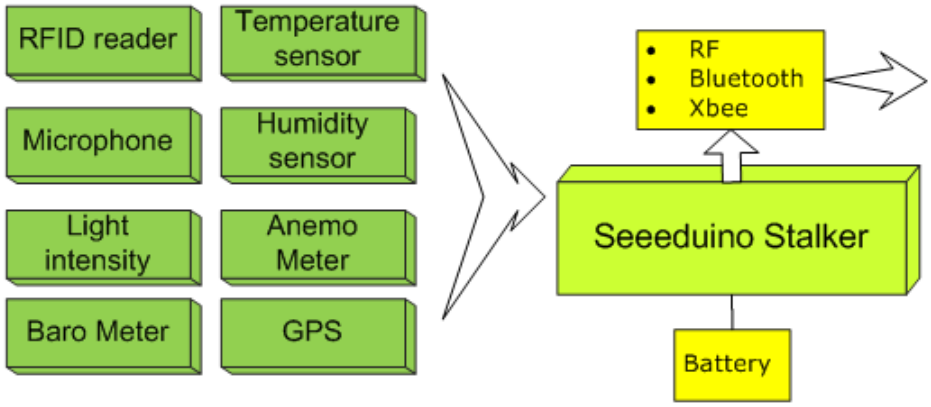
Powerful shields designed for data logging



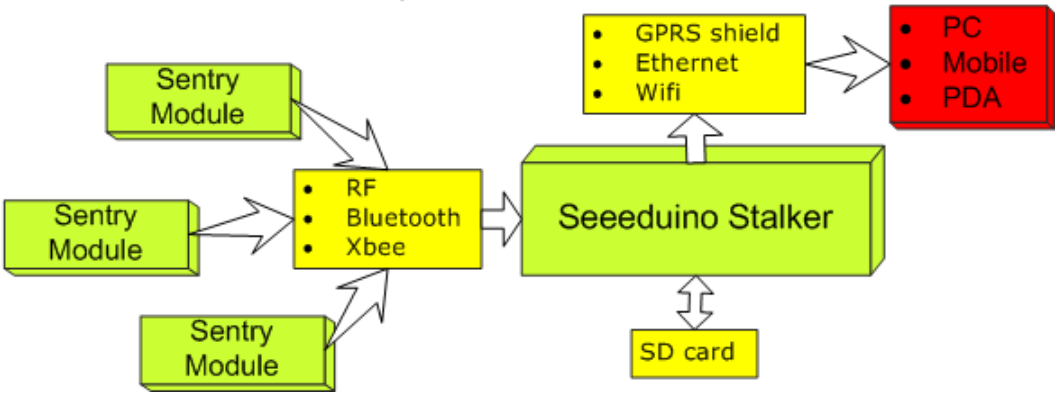
A minimal stand alone GPS logger - Log file from SD card



Sentry module - Stream data for real time monitoring



Smart Sensor network system - Bulk or Stream data upload via network



Demos

Programming on Stalker are made easy by Arduino and its libraries.

Demo1 - SD card usage

Using a FileLogger library to store all the data from GPSBee into SD Card.

```
#include "FileLogger.h"
#define Idle 0
#define Accept    1
#define Record    2

byte start[7]= { 'B','e','g','i','n',0x0d,0x0a};
byte buffer[128];
unsigned char result;
unsigned char state;
unsigned long ptr;


void setup(void)
{
   Serial.begin(9600);//start Uart with GPS
   result = FileLogger::append("data.log", start, 7);
   while(result)    result = FileLogger::append("data.log", start, 7);
}

void loop(void)
{
   switch(state)
   {
   case Idle: // just idle
      if (Serial.available()>0) state=1;
      break;

   case Accept: // receive data from GPS and save in buffer
      if (Serial.available()>0)
      {
         buffer[ptr]=Serial.read();
         ptr++;
         if(ptr>127)// if more than 128Byte in buffer
         {
            state=Record;
         }
      }
      else
```

```
    { // receive end
      state=Record;
    }
    break;

  case Record:// log the data in buffer into SD card
    result = FileLogger::append("data.log", buffer, ptr);
    if (result==0)
    {
      ptr=0;
      state=Idle;
    }
    break;

  default:
    state=Idle;
    break;

  }

}
```

Demo 2 – Save data with timestamp

On board RTC module could be used for adding timestamp to data sensed. A demo below illustrates storing sensor data to SD card with time info.

```
#include "FileLogger.h"
#include "DS1307.h"
#include <WProgram.h>
#include <Wire.h>

#define Timing 0
#define Accept  1
#define Record  2


byte start[7]= { 'B','e','g','i','n',0x0D,0x0A};
byte buffer[20];
int temp;
byte ASCII[10]={'0','1','2','3','4','5','6','7','8','9'};
unsigned char result;
unsigned char state;
int time=0;
int oldtime=0;
```

```
void setup(void)
{
  result = FileLogger::append("data.log", start, 7);//Initial the SD Card
  while(result)    result = FileLogger::append("data.log", start, 7);
  RTC.stop();
  RTC.set(DS1307_MIN,30);         //set the minutes
  RTC.set(DS1307_HR,10);           //set the hours
  RTC.set(DS1307_DATE,22);          //set the date
  RTC.set(DS1307_MTH,12);           //set the month
  RTC.set(DS1307_YR,9);            //set the year
  RTC.start();
}

void loop(void)
{
  switch(state)
  {
  case Timing:

          time=RTC.get(DS1307_SEC,true);
          delay(200);
          if(time!=oldtime)
          {
            oldtime=time;
            temp=RTC.get(DS1307_MTH,false);
            buffer[0]=ASCII[(temp/10)];
            buffer[1]=ASCII[(temp%10)];
            buffer[2]='-';
            temp=RTC.get(DS1307_DATE,false);
            buffer[3]=ASCII[(temp/10)];
            buffer[4]=ASCII[(temp%10)];
            buffer[5]='-';
            temp=RTC.get(DS1307_HR,false);
            buffer[6]=ASCII[(temp/10)];
            buffer[7]=ASCII[(temp%10)];
            buffer[8]='-';
            temp=RTC.get(DS1307_MIN,false);
            buffer[9]=ASCII[(temp/10)];
            buffer[10]=ASCII[(temp%10)];
            buffer[11]='-';
            //temp=RTC.get(DS1307_SEC,false);
            buffer[12]=ASCII[(time/10)];
            buffer[13]=ASCII[(time%10)];
            buffer[14]=':';
            state=Accept;
```

```
            }
        break;

    case Accept:
        temp=analogRead(0);
        buffer[15]=ASCII[(temp/100)];
        buffer[16]=ASCII[((temp%100)/10)];
        buffer[17]=ASCII[(temp%10)];
        buffer[18]=0x0D;
        buffer[19]=0x0A;
        state=Record;
        break;

    case Record:
        result = FileLogger::append("data.log", buffer, 20);
        if (result==0)
        {
            state=Timing;
        }
        break;

    default:
        state=Timing;
        break;

    }

}
```

Demo 3 Work as Arduino shield

Work as a datalogger shield , receive the data from Arduino via IIC and record them into the SD card(The Arduino use the wire master_write example code).

```
#include "FileLogger.h"
#include <Wire.h>


byte start[7]= { 'B','e','g','i','n',0x0D,0x0A};
unsigned char buffer[10];
unsigned char result;
unsigned char state;
```

```
void setup()
{
  result = FileLogger::append("data.log", start, 7);//Initial the SD Card
  while(result)    result = FileLogger::append("data.log", start, 7);
  Wire.begin(4);                        // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event

}

void loop()
{



}

void receiveEvent(int howMany)
{
  unsigned char i=0;
  while(Wire.available()>0)
  {
    buffer[i] = Wire.receive(); // receive byte as a character
    i++;
  }

  result = FileLogger::append("data.log", buffer, i);
  while(result)    result = FileLogger::append("data.log", start, 7);

}
```

Revision History

Rev.	Descriptions	Release date
v1.0b	Initial public release	12/23/2009