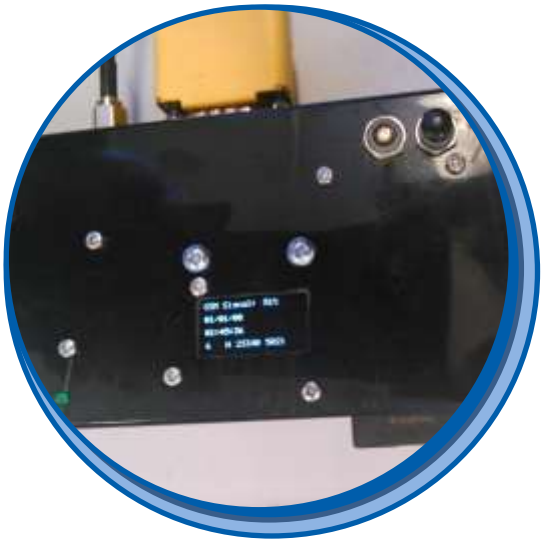
UNIVERSITY OF EXETER

Design of an Arduino-based Datalogger with MODBUS and GSM Functionality

CENTRE FOR ENERGY AND THE ENVIRONMENT

Software Document 41

February 2019



Report Name:	Design of an Arduino-based Datalogger with MODBUS and GSM Functionality
Author(s):	T.A. Mitchell
Report Number:	Software Document 41
Publication Date:	February 2019

CENTRE FOR ENERGY AND THE ENVIRONMENT

UNIVERSITY OF EXETER,
HOPE HALL,
PRINCE OF WALES ROAD,
EXETER,
EX4 4PL

T: 01392 724143
W: www.exeter.ac.uk/cee
E: t.a.mitchel@exeter.ac.uk

Rev. No.	*Comments*	*Approved By*	*Date*
0	Final Draft	-	-

CONTENTS

Management Summary

This report documents the latest development of a datalogger based around an Arduino microcontroller. It combines the functionality of previous designs to read values from digital meters using the MODBUS protocol and to transmit data to a fileserver via the mobile phone network.

The design is based around an Arduino Mega 2560, which provides significantly more memory for the program, variables and non-volatile storage than the Arduino Nano used in previous designs, and also has a greater number of inputs that could potentially be used to increase the number of meters or other sensors being monitored.

1. Introduction

This report documents the design and construction of a low-cost datalogger that both reads data from a MODBUS network and transmits data to a fileserver via the moble phone network. It has been developed from the MODBUS datalogger described in Software Document 39 [1], and the GSM datalogger described in Software Document 40 [2]. The MODBUS protocol is widely used by electronic metering devices including electricity and heat meters. The Global System for Mobile communications (GSM) allows communications over a "2G" mobile phone network. Both designs were based around the Arduino Nano microcontroller.

Most of the components are identical to those used in the previous designs, but the microcontroller that is programmed to interact with the other components has been upgraded from an Arduino Nano to an Arduino Mega 2560. The GSM datalogger utilised nearly all of the available program, random access and non-volatile memory resources of the Arduino Nano and the increased resources provided by the Arduino Mega enabled the previous designs to be combined relatively simply, as well as providing significant additional resources for future modifications (e.g. to record data from other devices such as I2C-based environmental sensors or XBee wireless communication with sensor outstations). Table 1 compares the resources available on the Arduino Nano and Mega, and the resources utilised by the MODBUS GSM datalogger.

Table 1. Resources available on the Arduino Nano and Mega, and those utilised by the MODBUS GSM datalogger.

Resource	Arduino Nano	Arduino Mega 2560	Resources in Use
Flash (program) memory	32 kB [$]	256 kB [$]	34 kB
SRAM (data) memory	2 kB	8 kB	2.1 kB [%]
EEPROM (non-volatile data) memory	1 kB	4 kB	1.1 kB
Digital In/Outputs	20 *	70 [#]	19
Analogue Inputs	8	16	0
Serial Ports	1 [†]	4 [†]	2 [@]

[$] This includes memory used by the operating system and bootloader, approximately 0.5 kB for the Arduino Nano and 8 kB for the Arduino Mega 2560.

[%] For global variables. Further memory is allocated dynamically to local variables.

[*] Six are shared with the analogue inputs.

[#] 16 are shared with the analogue inputs.

[†] Shared with the digital in/outputs (each serial port uses two digital in/outputs).

[@] One further serial port (shared with the USB interface) is used to program the unit and for debugging using terminal software.

2. Build Notes

The design approach and hardware have been described in previous documents [1], [2]. These notes document specific design changes and challenges that were encountered.

- **Serial Communications**: the availability of multiple serial ports on the Arduino Mega 2560 microcontroller allowed the *SoftwareSerial* library previously utilised to facilitate communication with the GSM modem on any digital pins to be dispensed with. In any case, the Arduino Mega 2560 only supports use of SoftwareSerial communications on a small subset of digital pins, as "change interrupt" support is required on the receive pin, which is only available on digital inputs 10 to 15, 50 to 53 and 62 (A8) to 69 (A15). MODBUS communication (via an RS-485 interface converter) has been re-allocated to serial port 2 (digital pins 16 and 17) in the `modbus_configure` function call. This frees up the default serial port (`Serial`) on digital pins 0 and 1, allowing communication via USB for debugging purposes (the USB port shares

this serial port). GSM modem communications have been re-allocated to serial port 1 (digital pins 18 and 19): the port object named `gsm` previously created using the `SoftwareSerial` command has been dispensed with and data are read from and written to `Serial1` directly.

- **Memory usage**: In addition to the code specific to the design, the declared libraries utilise much additional flash memory. Significant amounts of SRAM are also consumed by the use of peripherals (e.g. the SD card library creates a 512 byte write buffer).
- **GSM Modem**: it was planned to use a SIM800C module in place of the SIM800L due to alleged improved reliability, but the lack of a reset pin on the module[1] obtained and uncertainty over operating signal levels led to the SIM800L being retained.
- **SPI and I2C pins**: the pins used for SPI communication with the SD card and I2C communication with the display and real-time clock differ between the Arduino Nano and the Arduino Mega 2560. See Appendix 1.
- **Changes to the software** include uncommenting MODBUS-specific commands from the GSM datalogger software, changing the configuration file to include the settings required for both pulse count logging and MODBUS logging, similar changes to datafiles used to store logged data, changes as noted above for serial communications, and the addition of MODBUS data readouts to the user-selectable on-screen data display.
- **The Data Input Connection** via the 9-pin D connector has added the four pulse inputs to the previously unused pins 3, 4, 7 and 8. With the expanded capabilities of the Arduino Mega 2560 it would be easy to add support for a greater number of pulse counting inputs simply by increasing the size of the following arrays: `PinIn`, `OldInputState`, `PulseWeight`, `PulseReading`, `PulseName`, `PrevPulseTime` and `NewPulse`, (and setting the initial values appropriately where required in the variable declaration: to zero, except in the case of `PinIn` where the digital pin should be specified for each input). An additional connector would be required to provide access to the appropriate pins.
- **LEDs**: are now common cathode so the output state has been reversed in code, the anode pin declaration removed and the cathodes connected to ground.
- **RS485 Interface**: The A and B connections appear to be transposed from the usual convention on the RS485 to TTL interface board, this was not noted in the previous documentation.
- **Enclosure**: the enclosure is relatively cramped and a larger enclosure would be beneficial for future designs (and allow further connector(s) to be provided for external access to further digital inputs, as well as the SPI and I2C buses which may be useful for communications with other sensors).

3. Operation Notes

Operation is similar to the two previous designs. Settings are read from a file *gsmmod.cfg* (Appendix 4); a sample file is created if the datalogger is started with an SD card inserted that soes not include a file of this name.

Power is supplied from a 9 volt 2.5 amp supply to the **circular** 2.1mm power connector: the rectangular connector supplies the Arduino Mega 2560 directly (and thence the display, real-time clock and SD card interface, but not the GSM modem) and should not normally be used. Likewise, the USB connector will not supply power to the GSM modem.

[1] A power key (PWK) pin was available that could potentially be used to reset the unit, but would then need an initial connection of this pin to ground to boot up the unit.

The tricolour LED has been replaced by red and green LEDs, so any reference to a yellow LED (indicating user input required, e.g. to set the clock or start logging) is indicated by both LEDs being illuminated.

Two buttons are provided: the "sawn off" one functions as a recessed Reset button.

References

1. ***Design of a Datalogger for MODBUS.*** SWEEG Software Document 39. 2018, T.A.Mitchell.
2. ***Design of an Arduino-based GSM Datalogger.*** SWEEG Software Document 40. 2019, T.A.Mitchell.
3. ***Serial Monitor Deluxe.*** https://www.idogendel.com/en/products/serial-monitor-deluxe, last accessed 19/12/2018.

APPENDIX 1 PIN USAGE

Table 2. Pin usage on the Arduino and external connector. Unlisted pins are unused.

Arduino Pin	D Connector Pin	Allocation	Note
D0	-	Spare	Used for built-in USB interface for echoing data to computer during testing
D1	-	Spare	Used for built-in USB interface for echoing data to computer during testing
D4	-	GSM Reset	Hold Low to reset GSM module
D5	-	Button	Pushbutton, connect between D5 and Ground
D6	3	Pulse input 1	
D7	4	Pulse Input 2	
D8	7	Pulse Input 3	
D9	8	Pulse Input 4	
D10	-	Spare	Possibly reserved by the SD card library as well as pin 53.
D11	-	Green LED anode	470Ω resistor.
D12	-	Red LED anode	470Ω resistor.
D14	-	RS485 Interface (DE + RE)	DE high to transmit and RE low to receive, so for half duplex comms can tie DE and RE together
D16 (TX2)	-	RS485 Interface (DI)	Data Transmitted to RS485
D17 (RX2)	-	RS485 Interface (RO)	Data Received from RS485
D18 (TX1)	-	GSM Modem	Transmit to modem
D19 (RX1)	-	GSM Modem	Receive from modem
D20 (I2C)		RTC & OLED (SDA)	RTC address is 0x68 and OLED is 0x3C
D21 (I2C)		RTC & OLED (SCL)	Addresses may vary between similar modules
D50 (SPI)		SD Card (MISO)	
D51 (SPI)		SD Card (MOSI)	
D52 (SPI)		SD Card (SCK)	
D53 (SPI)		SD Card (CS)	Can use other digital pins, but SD.h always reserves pin D53
Vin	9	7-12 V DC Supply +ve	Power supply to device (to power socket centre pin), and available on a terminal for use to power meter pulse circuitry
5V	6	5 V supply from Arduino	Power to RTC, SD, OLED display, and available on a terminal for use to power meter pulse circuitry
RST		Arduino Reset	Wired to external pushbutton, momentarily ground to reset
GND	5	To RTC, SD, OLED, LEDs, buttons, power supply -ve contact	Available on a terminal for use to power meter pulse circuitry
-	1	RS485 B+	For Modbus A and B markings transposed on circuit board
-	2	RS485 A-	For Modbus A and B markings transposed on circuit board

APPENDIX 2 CODE VERSION HISTORY

Table 3 Version history: main code.

Version	Notes
003	First tested version

APPENDIX 3 GSM ACCOUNT DETAILS

The system has been tested with a GiffGaff SIM card

Username (for online account management): CEEArduino2

Password (for online account management): Ardu1n0Mega

Telephone number: 07541 954536

The account must be used once every six months to keep it active, either by transmitting data or a text message.

To transmit a text message, load the code listed below to the Arduino. With the Arduino connected to the PC via USB open Serial Monitor Deluxe [3] (or another terminal program that allows transmission of non-printable ASCII characters) and enter the following commands:

`AT+CMGF=1`

`AT+CMGS="01234 56789"` *(replacing the number with a valid phone number)*

Type text of message

Enter ASCII character 26 to end (enter \26\ in Serial Monitor Deluxe)

Code Listing (saved in \\cee-nas\Common\Admin\Equipment\datalogging\arduino\Arduino GSM Logger\Modbus version\GSM_Mega)

```
//Comms tests for GSM modem on Arduino Mega.
//GSM comms on hardware serial port 1, not software serial

void setup()
{
 Serial.begin(9600); //For serial monitor on PC via USB
 Serial1.begin(4800); //for GSM
}

void loop()
{
 if(Serial1.available())
 {
  Serial.println(Serial1.readString());
 }
 if(Serial.available())
 {
  Serial1.println(Serial.readString());
 }
}
```

APPENDIX 4 EXAMPLE CONFIGURATION FILE

```
Update Time (N/Y): N
New Time (YYMMDDHHMMSS): 181211125800
Wait for keypress to start (N/Y): N
Max Pulse Input No (1/2/3/4): 2
Pulse Weight 1 (1-65535 units per pulse): 1
Pulse Weight 2 (1-65535 units per pulse): 1
Pulse Input 1 Name (1 to 8 characters): Input 1
Pulse Input 2 Name (1 to 8 characters): Input 2
Pulse Debounce (1-255ms): 50
Log Rate (1-65535 S): 300
Upload Rate (H/D): H
Multi Column Format (N/Y): Y
Specify Starting Readings (N/Y): N
Meter Reading Input 1 (0-999999): 5678
Meter Reading Input 2 (0-999999): 89745
Modbus Baud Rate: 9600
Modbus Parity (N/O/E): O
Modbus Stop Bits (1/2): 1
Number of Modbus Registers: 4
Registers to Log (Name,Slave ID,Type (C/S/I/H),Reg No):
Modbus1,6,H,20483
Modbus2,6,H,23297
Modbus3,6,H,23309
Modbus4,6,H,23340
Length of signal strength command: 6
Signal Strength Command: AT+CSQ
Signal Strength Response Data Delimiter: :
Signal Strength Response Length: 3
Number of Communication Test Commands: 3
Length of First Communication Test Command: 8
First Communication Test Command: AT+CREG?
Length of First Communication Test Command Valid Response: 9
First Communication Test Command Valid Response: +CREG:0,1
Length of Second Communication Test Command: 12
Second Communication Test Command: AT+SAPBR=1,1
Length of Second Communication Test Command Valid Response: 2
Second Communication Test Command Valid Response: OK
Length of Third Communication Test Command: 11
Third Communication Test Command: AT+FTPSTATE
Length of Third Communication Test Command Valid Response: 11
Third Communication Test Command Valid Response: +FTPSTATE:0
Number of GSM Initialisation Commands: 6
Length of First GSM Initialisation Command: 9
First GSM Initialisation Command: AT+CFUN=1
Length of First GSM Initialisation Command Valid Response: 2
First GSM Initialisation Command Valid Response: OK
Length of Second GSM Initialisation Command: 10
Second GSM Initialisation Command: AT+CGATT=1
Length of Second GSM Initialisation Command Valid Response: 2
Second GSM Initialisation Command Valid Response: OK
Length of Third GSM Initialisation Command: 29
Third GSM Initialisation Command: AT+SAPBR=3,1,"CONTYPE","GPRS"
Length of Third GSM Initialisation Command Valid Response: 2
Third GSM Initialisation Command Valid Response: OK
Length of Fourth GSM Initialisation Command: 33
Fourth GSM Initialisation Command: AT+SAPBR=3,1,"APN","giffgaff.com"
Length of Fourth GSM Initialisation Command Valid Response: 2
Fourth GSM Initialisation Command Valid Response: OK
Length of Fifth GSM Initialisation Command: 30
Fifth GSM Initialisation Command: AT+SAPBR=3,1,"USER","giffgaff"
Length of Fifth GSM Initialisation Command Valid Response: 2
Fifth GSM Initialisation Command Valid Response: OK
Length of Sixth GSM Initialisation Command: 12
Sixth GSM Initialisation Command: AT+SAPBR=1,1
Length of Sixth GSM Initialisation Command Valid Response: 2
Sixth GSM Initialisation Command Valid Response: OK
Number of FTP Initialisation Commands: 5
Length of First FTP Initialisation Command: 11
First FTP Initialisation Command: AT+FTPCID=1
Length of First FTP Initialisation Command Valid Response: 2
First FTP Initialisation Command Valid Response: OK
Length of Second FTP Initialisation Command: 29
Second FTP Initialisation Command: AT+FTPSERV="web-ftp.ex.ac.uk"
Length of Second FTP Initialisation Command Valid Response: 2
Second FTP Initialisation Command Valid Response: OK
Length of Third FTP Initialisation Command: 17
Third FTP Initialisation Command: AT+FTPUN="web684"
Length of Third FTP Initialisation Command Valid Response: 2
Third FTP Initialisation Command Valid Response: OK
Length of Fourth FTP Initialisation Command: 19
Fourth FTP Initialisation Command: AT+FTPPW="Util-4EX"
Length of Fourth FTP Initialisation Command Valid Response: 2
Fourth FTP Initialisation Command Valid Response: OK
Length of Fifth FTP Initialisation Command: 54
Fifth FTP Initialisation Command: AT+FTPPUTPATH="/services.exeter.ac.uk/utilities/test/"
Length of fifth FTP Initialisation Command Valid Response: 2
Fifth FTP Initialisation Command Valid Response: OK
Length of GSM Set Filename command (to start of filename): 14
GSM Set filename command (to start of filename): AT+FTPPUTNAME=
```

```
Length of GSM Set filename command Valid Response: 2
GSM Set filename command Valid Response: OK
Length of GSM Start FTP command: 11
GSM Start FTP command: AT+FTPPUT=1
Length of GSM Start FTP command Valid Response (excluding data buffer length returned): 19
GSM Start FTP command Valid Response (excluding data buffer length returned): OK

+FTPPUT: 1,1,
Length of GSM Request FTP Data send command: 12
GSM Request FTP Data send command (exc. Data length): AT+FTPPUT=2,
Length of GSM Request FTP Data send command Valid Response (exc. Data length): 11
GSM Request FTP Data send command Valid Response (exc. Data length): +FTPPUT: 2,
Length of GSM End FTP Data send command: 13
GSM End FTP Data send command: AT+FTPPUT=2,0
Length of GSM End FTP Data send command Valid Response: 17
GSM End FTP Data send command: OK

+FTPPUT: 1,
Length of GSM End FTP Session command: 12
GSM End FTP Session command: AT+SAPBR=0,1
Length of GSM End FTP Session command Valid Response: 2
GSM End FTP Session command: OK
```