Low power solar PV logger

For a lighting system for a hut in Scotland used by various groups

Adapted from https://github.com/chickey/RS485-WiFi-EPEver

Last Updated 29/01/24

Why?- The problem

Because :-

- Winter solar irradiation is low.
- Occupancy has increased.
- Maybe using lights excessively.
- Verbal reporting of battery level by visitors is not consistent.
- Lithium Battery charge level on EPever charge controller is inaccurate (New Nov23)

What? – The Solution

- Log last date when battery fully charged
- Log subsequent energy use, solar generation and calculate remaining battery capacity. Known a 'Columb counting'
- Display using GUI for non-technical users.
- Graphic showing use against time (optional)
- Data upload for analysis.

Design Criteria

- Burden: the amount of power consumed by the logger itself must be minimised.
- Operation and UI must be straight-forward.

Keeping it simple

- Modify code written by Chickey… does most of what we need.
- ESP8266 in deep sleep and woken approximately every 40 Seconds using Timer-RESET

  - Read Epever parameters using MODBUS save to FRAM and shutdown

  - IF RTC time is midday then Epever LOAD OFF

  - IF load transition OFF- ON has occurred wake+wi-fi to allow user interaction

  - Display parameters on 'capture-portal', (No network available or req'd)

  - Download CSV log files using web GUI

  - Graph previous week/months load on GUI (optional)

  - Configuration? password protected (hard coded)

https://forum.arduino.cc/t/file-download-with-arduino-web-server/160140/9

-Wireless doorbell for remote switching of the Epever Load 'Master Lights ON/OFF'

- also provides a nice box

Detailed Design

Refer to circuit diagram EPEVER-ESP8266 -12F683 datalogger

MODBUS input interface ST3845 EIA-485- half duplex low power transceiver. Line polarisation and termination R3,4 and 5 R1 and R2 bias the device into shutdown when the ESP8266 is in deep sleep.

The ESP8266 microcontroller has integrated wi-fi and TCP/IP networking software

Minimising burden current is important. Just 10mA of constant current draw will deplete the 12 Volt 220ah storage battery by ~ 3% a month. Design will be < 100µA

The ESP8266 typically draws 70+ mA in normal operation, deep sleep approx. 20uA.[REF 01.]

Solar irradiation and light switches change slowly, 40 seconds sleep between data reads

Sampling rate determined by need to poll EPever panel load pushbutton and flash OK LED.

There are ESP8266 hardware design constraints:

- Wake through pin interrupt is not possible from deep sleep.
- Sleep cycle puts the ESP through reset, variables stored in NV memory.
- The ESP8266 internal sleep timer is only 5% accurate.

I did consider other devices but:

- Integrated wi-fi -simplifies hardware.
- Prewritten code and application notes available
- Interrupt work-around using 8 bit 12F683 adds functionality

Microchip 12F683:

- Passes through the ESP sleep timeout from D5 to reset.
- OR in the wireless doorbell control to reset line.
- Internal WDT to reset the ESP8266 after 180 seconds of inactivity.
- Flashes an LED to indicate AOK

Code ATM loops and polls. 32KHz CLK typically I_{DD} typically 20µA, Max 30 µA

Re-write to sleep the controller with IOC on inputs would reduce to 2-4 µA

As the Epever solar controller itself takes 14 mA that idea is just a maybe..

NV memory

I did consider an SD card, simply swap to recover data, however they are slow, and require additional power management hardware as they take 10's of mA when idle. Not enough spare ESP pins to implement. *And we just don't need to record very much data..*

Introducing FRAM: 32K Bytes, fast, simple implementation, 15 µA standby current, 10^{14} read/write cycles.

So let us run through the basic programme flow..

- ESP8266 reads EPever parameters and stores to FRAM (or ESP RTC memory)
- Sleep timer set > deep sleep > Sleep timeout +40 Seconds D5 set low. 12F683 resets ESP8266
- Accumulate parameter values Prev + Current
- Read Epever RTC. IF one hour has passed save hours parameters to FRAM database
- Read Epever RTC. IF one day has passed save days parameters to FRAM database
-

IF 12F683 wakes ESP8266 due to bell push input

- Flags this input GP0 > D5 [active low]
- ESP setup code toggles EPever load
- IF load switch state OFF>ON then wake wifi and go into main loop
- ESP Loads GUI and starts programme loop
- Await UI
- If UI not present after timeout revert to sleep mode

- 12F683 resets it's WDT while ESP in loop flagged by GP2 low

FRAM logging

With just 32K bytes of FRAM we need to be a bit smart about how we save data.

- Accumulate data hourly, e.g. solar irradiation of constant 20 watts for one hour: 20 Wh
  30 mins of 10 Watts plus 30 mins of 20 Watts: 15 Wh
- Times and dates saved as packed BCD 16 bits for hh/dd/mm/yy
- If we have a null record, no load and no solar power for an hour then skip saving
  Knowing there are 24 hours in a day we can fill in gaps if plotting data
- Record cumulative results for each day
- Write hourly and daily results to fixed size memory maps, overwrite oldest record.

Hourly Record

Parameter	Range	Notes	Bits	Cuml.Bytes
hh/dd			8	1
mm/yy			8	2
Solar Volts	0-40 Volts	0.5 V res mask msb for flg_0	7	3
Solar Energy	0-100 Watt/h	1 wh res mask msb for flg_1	7	4
Load Energy	0-100 Watt/h	1 wh res mask msb for flg_2	7	5
Battery temp	-20 to +40	0.5 C resolution	8	6
Load ON	flg_0		1	
Under temp	flg_1	Lithium protection on =	1	
Battery Full	flg_2		1	

Daily Record

Parameter	Value	range	Comment	Bits	Bytes
dd/mm				8	1
flgs/yy				8	2
Peak Solar power	watts	0-120		8	3
Solar energy today	w/h	0-1024	mask off four flag bits from MSB	10	5
Load energy (total day)	w/h	0-254		8	6
Battery Volts (now)	V	10.7-15		8	7
Calculated Capacity		0-100%	Columbs counted from last full	8	8
Load ON (today)	BIN	True/False	flgs_0	1	
Battery Full (today)	BIN	True/False	flgs_1 (>=14.2 V + no current)	1	
Under temp today	BIN	True/False	flgs_2	1	
TBA	BIN	True/False	flgs_3	1	
Min Temp (today)	degC	-20 to +40C	-20 to +40C @ 0.25 resolution	8	9
MaxTemp (today)	degC	-20 to +40C	-20 to +40C @ 0.25 resolution	8	10

Hourly record,

6 bytes/h, assume 12 hours of records a day, gives 72 bytes a day,.. 504 weekly and 15K monthly

Daily record

10 bytes of daily data, 3650 a year… allocating 8K - more than two years of data

8 K free to save variables during sleep period, indexing records

Accumulating data

Sample: Temperature, Solar and Load power every 40 seconds

Every minute (rollover on EPever RTC) add instantaneous power to total e.g. 10 Watts Increment sample number

At end of day.. w/h = Cumulative / number of samples * 24 100 watts for 12 hours = 1200, samples 1440 = 4 BYTES

Links

https://community.platformio.org/t/correct-settings-for-esp8266-d1-mini/30681

Program design

Andreas Spiess

Using RTC memory

https://www.youtube.com/watch?v=r-hEOL007nw&t=64s

ESP deep sleep

https://www.youtube.com/watch?v=r75MrWIVIw4

Writing to EEPROM

ESP8266 have 4K bytes of EEPROM

https://circuits4you.com/2016/12/16/esp8266-internal-eeprom-arduino/

(512 Bytes of NVRAM)

https://www.aranacorp.com/en/using-the-eeprom-with-the-esp8266/#:~:text=The%20EEPROM%20of%20the%20ESP8266%20has%20a%20size%20of%204kB.

Once daily recording parameters efficiently as packed BCD and scaled values requires 15 bytes

Programing write address as

https://www.arduino.cc/reference/en/libraries/osfs/

Hardware MH-ET-LIVE

https://riot-os.readthedocs.io/generated/group__boards__esp32__mh-et-live-minikit.html

https://doc.riot-os.org/group__boards__esp32__mh-et-live-minikit.html

References

REF_01: https://www.scirp.org/journal/paperinformation?paperid=90552

FRAM

https://www.infineon.com/dgdl/Infineon-FM24W256_256_KBIT_(32K_X_8)_SERIAL_(I2C)_F-RAM-DataSheet-v13_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ec9dd494223