Home Alone Monitor

Project Description

Work in progress

Peter K. Boxler

January 2020

Version 2

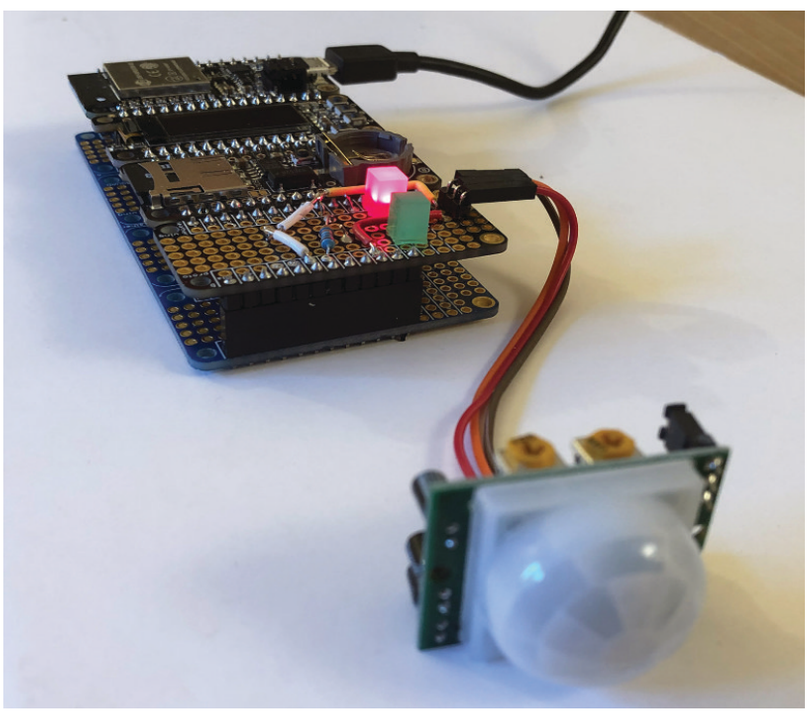
Inhaltsverzeichnis

1. Abstract

This documentation presents a small electronic gadget called Home Alone. Some folks concerned with the well being of elderly people living alone could use it to keep track of the movements of their loved ones. This gadget detects, counts and reports movements in its surroundings. For privacy reasons no camera is used, a simple PIR sensor (Pyroelectric Infrared Sensor) detects movement. Movements (or the absence of it) are reported to the cloud. Concerned party (or parties) can inspect the data on a website.

2. Introduction

Recently a friend told a story of an elderly woman beeing found dead by her daughter. Luckily, the daughter came by just one day after the old lady passed away. Everybody has heard of similar stories. That got me thinking and I knew it would be rather easy to build a gadget helping to prevent such stories. However, a couple of days later I discovered that (here in Switzerland at least) one can buy electronic devices that detect movement and they are connected to the landline. So I dropped the plan to build such a device.
A couple of weeks later, however, I came across Ralph Bacons Maker Videos on YouTube and bingo, he describes just such a devices in one of his videos.
Since most of the parts needed to build such a gadget were lying around in my lab I decided to give it a try. Here is what came out of it. It is just a prototype as of now.



Home Alone Monitor Prototype

3. What does it do

Let's assume we have an elderly uncle called Dagobert. He lives alone in an appartement in a city a couple of hours away. He still can live on his own, is even able to walk to the shop or to the city park and he can handle a phone. His neighbours keep an eye on him and they promised to call us if they have not seen him in days.
So this is where this device comes in handy: it detects movements (in the appartement) and counts them. Movement counts are reported to the cloud in defined intervalls (every hour or so) and can be inspected on a smartphone anywhere in the world. The provider Thingspeak is used for that. Twice a day (morning and evening) a push-message is sent to a smartphone reporting the movement count during the day. Provider Pushover.net is used to do that. So even if we forget to check the Thingspeak website we will have a message on our phone. All of these threshold an intervalls are configurable - the configfile resides on a micro-sd card.. This scheme gives us some reassurance that all is well with uncle Dagobert (or not, see discussion at the end of this document). Between 11 pm (23 Uhr) and 8 am (8 Uhr) there is no reporting (also configurable)..

4. Parts used

I love the Adadfruit Feather series of boards and therefore decided to build with what I already had. Ralph Bacon device uses an ESP8266 but I was keen on using the more modern ESP32.
So I assembled these Feather boards for my device:

- Feather Huzzah ESP32, Product ID: 3405
- Featherwing OLED 128x32, Product ID: 4091
- Adalogger Featherwing, Product Id: 2922
- Featherwing Proto, Product ID: 2884
- Adafruit Quad Side-by-side, Product ID: 4245
- PIR Sensor, Product ID: 189
- Two LED, resistors
- Micro-SD card (any size will do)
- Power Supply 5 Volt 1 A

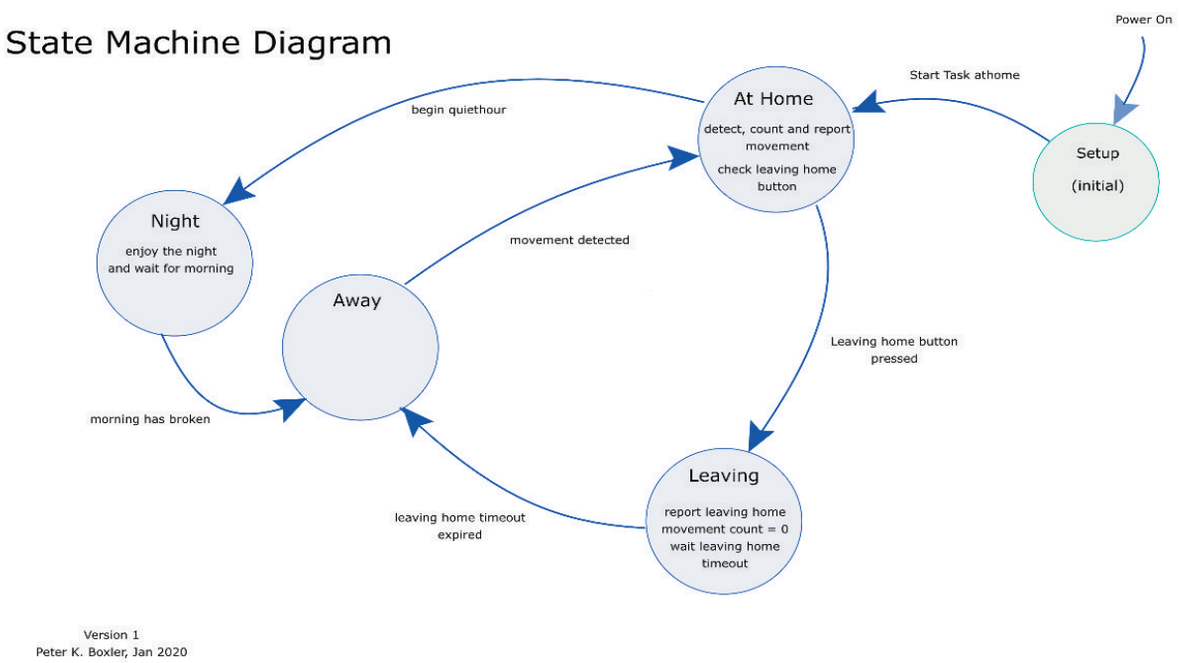
5. Software

The nature of the problem calls for an implementation of a Finite State Machine. Uncle Dagobert, whose movements are to be detected/reported can either be at Home, Leaving the home or be away.
I choose to adopt Ralph Bacons idea that uncle Dagobert should press a button before leaving the apartement.
Frankly, however, I suspect that in most cases Dagobert simply forgets to do this. So there is a great deal of interpretation when one looks at the movement data reported to the cloud.

I also wantd to make use of the multitasking capabilities of the ESP32 chip.
I used the Arduino IDE to develop the code and for readabiltiy and beauty I arranged the code into a couple of modules. Ralphs code is hard to read - all in one long piece.

6. Finite State Machine

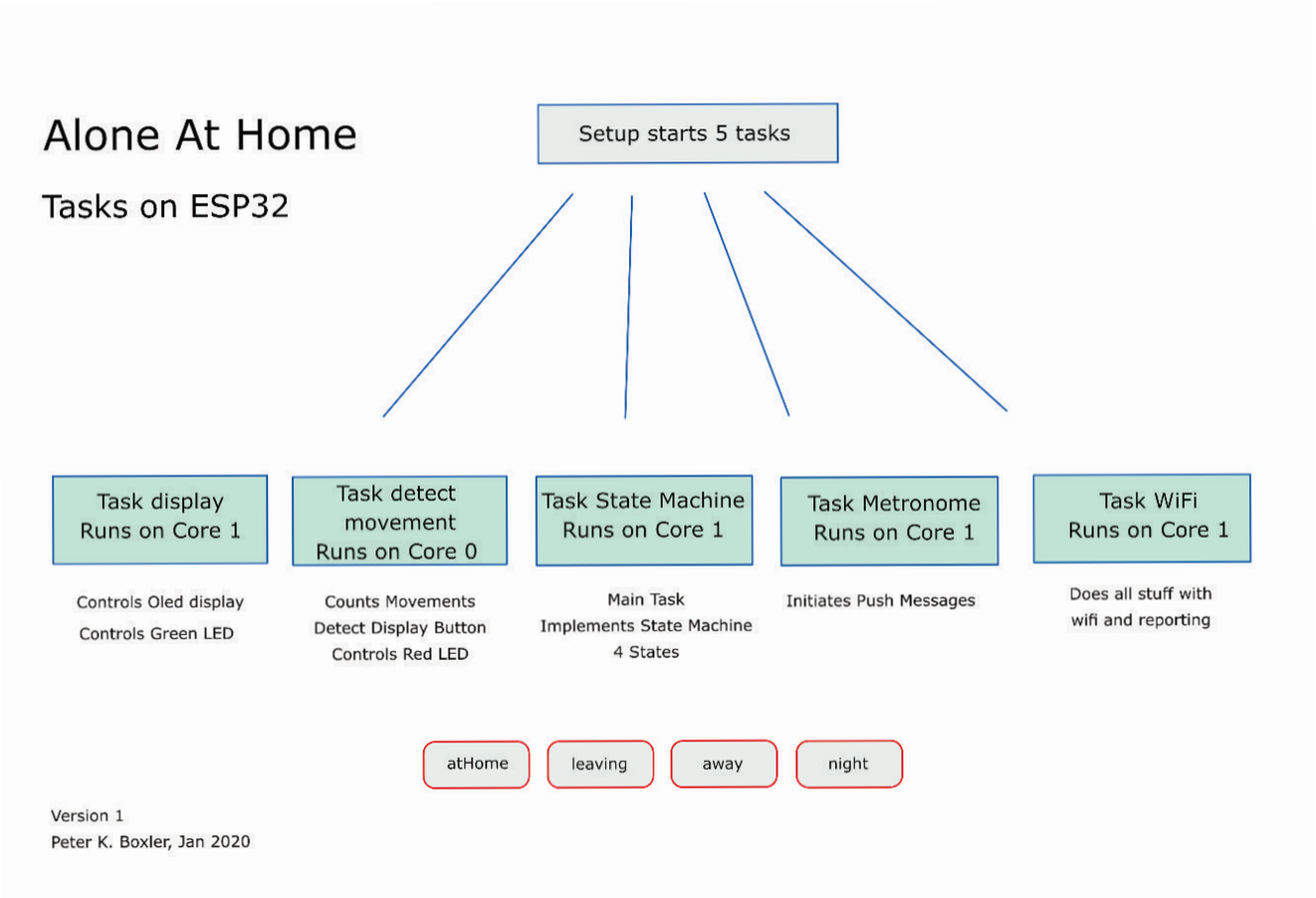
This the state diagram



State Diagramm of the application

7. Multitasking ESP32

This chip has two cores, runs with 240 Mhz clock an there is even a Real Time Operating System (RTS) on board. My code makes use of this capabilities and the setup function (arduino speak) starts 5 concurrently running tasks. See description.



ESP32 Tasks

7.1 The Tasks

Here is a list of tasks and what they do:

- Task detect movement: its job is to read the input pin where the pir sensor ist connected. As long as the pin is high the red led is on. At the end of the movement (pir signal goes low again) movement count_1 and count_2 are incremented and the red led is switched off.
- Task display: its job is to show data on the oled display if requested. Whenever the display is switched on the task will also switch off the display after n seconds to prevent burnout. It also checks the display button: if pressed the display is turned on. The values to be displayed are found in 5 global variables. A semaphore protected variable tells the task to switch on the display.
- Task state machine: this is the main task that implements the state machine.
- Task wifi: this task handles wifi connection, reporting to the cloud and the sending of push messages. It suspends itself after an action. It is resumed by one of the other tasks if one of them needs one of the actions to be performed. The wifi connection is disconnected for most of the time. Data ist passed to the task in a structure taht is protected by a semaphore.
  The configuration (config file) allows for two possible ssid. t
- Task clock: its job is to issue 'clock pulses' for several different events. With that scheme we only

need one code that deals with days, hours, change of days and years. At the moment, only 1 clock pulse is implemented (clock 1):

- 'clock pulse' for uplading movement count to the cloud. Whenever this clock is set to high by the clock task, the athome function (running in the statemachine task) does a reporting to the cloud.

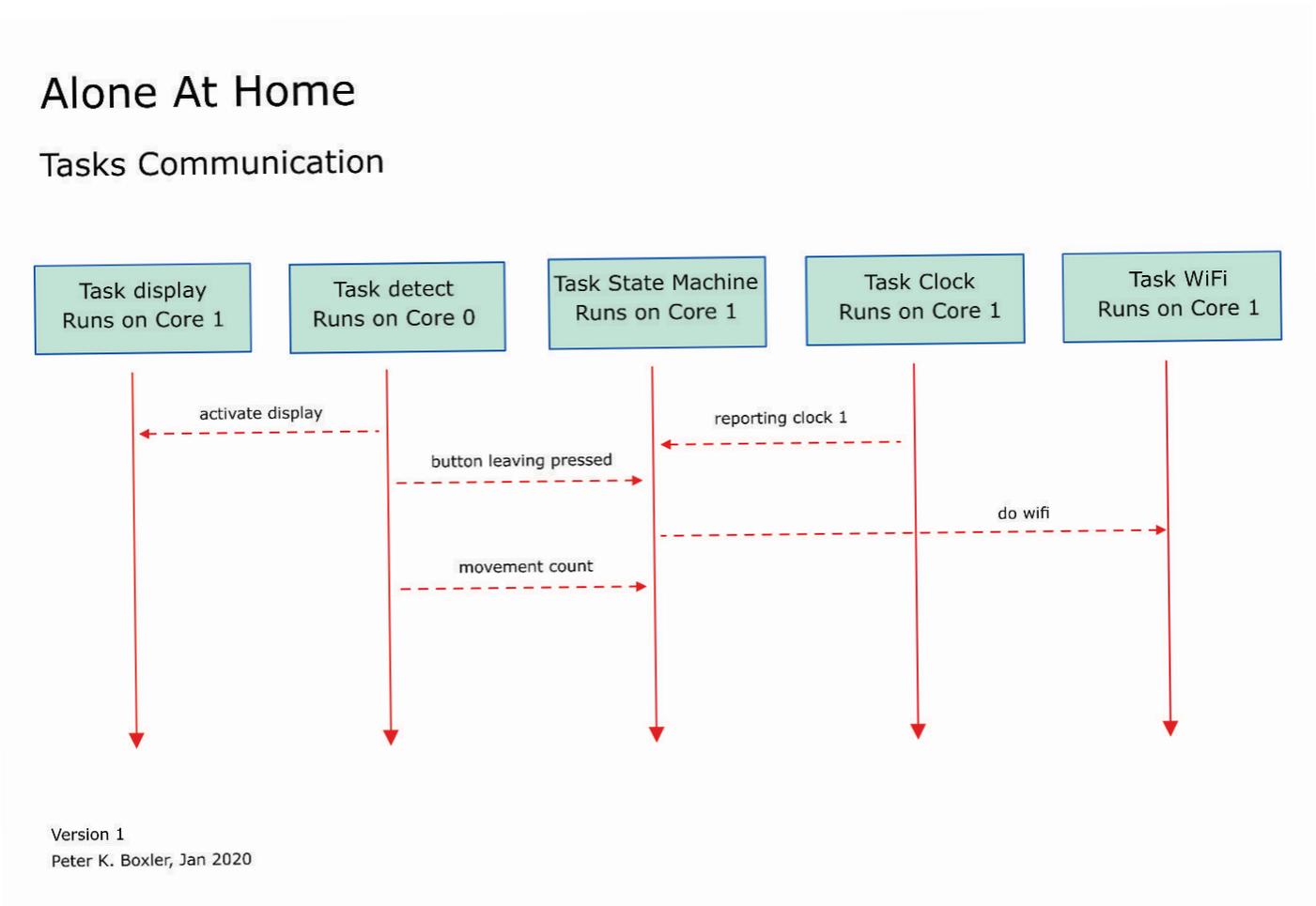
Of course 'clock pulses' is just a name for semaphore protected variables that are used as signals. All the other tasks simply query their relevant signal.

Note 1:

Note that the code that is run in setup also runs in a task in core 1 - after setup is done this task is **deleted**.

7.2 Task communication

Communcation is done by global variables that are protected by mux semaphores. Here is a diagram showing some of the communications



8. Thingspeak Channel

The rmovement reporting is done with Thingspeak.
The channel used is public an can be found here.:

Home Alone

9. Push Messages

This applicatiuoin uses push messages rather than mail. The provider pushover.net works just fine, is very cheap (about 5 $ for iOs plattform). on the iPad /iPhone an app is used to receive the push messages. Upon arrival of a message, the phone beeps and the message pops up an the lock screen.

10. Code

The code is available on GitHub.

Home alone on GitHub.

There are a number of modules:

- homealone.ino: this is the main module containig the setup function
- func_config.ino: contains the function loadconfig()
- func_div.ino: contains several functions
- task_clock.ino: code for clock task
- task_detect-ino: code for detect movement
- task_statemachine.ino: all code for the state machine
- task_wifi.ino: code to handle wifi
- images.h: contains the wifi image that is display on the oled display at startup

11. Config-File

The configfile contains all the configurable parameters. It is a json file.

12. Discussion, Conclusion

When uncle Dagobert shuffles around in his appartement and his WLAN is up we habe no problem: we can see that he is alive.
But what do we really know if no movements are reported to the cloud - this is the tricky part.

In that case we need to differentiate between:

- A) no more data points on the Thingspeak channel
- B) we see datapoints but reported movement count is zero

For A): we do not have any info about Dagobert himself, because:

- The WLAN might be dead in his appartement for a multitude of reasons unconnected with Dagobert himself.
- Dagobert might have tripped over the power cord and the device is dead,
- The ESP32 might have died but Dagobert is still alive and well,
- Thingsspeak as a company has folded or their computers have a serious problem.

For B): we know the device is still ok, but this should concern us, because:

- Dagobert might have died in his sleep last night,
- While shopping, he might have met this lovely old lady and they ran off together,
- Dagobert has left the appartement to go on a cruise.

We conclude that this device is nice and can help us to some degreee but....
All in all, we are happy we have it in place in Dagobert's appartement.

end of document