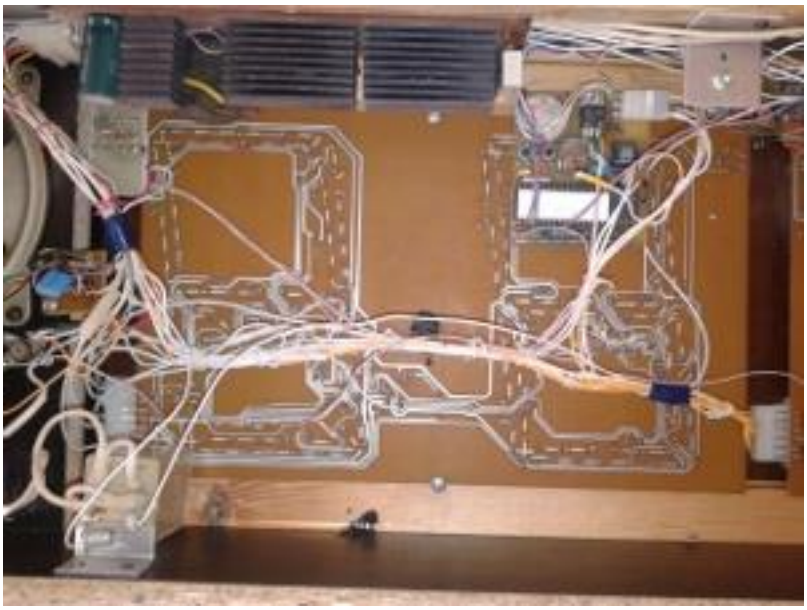
Доброго времени суток, уважаемые хабровчане. Не секрет, что одноплатные Linux компьютеры на базе SoC на сегодняшний день получили широкое распространение как среди любителей, так и среди более-менее профессиональных пользователей. Все больше и больше задач можно решить с помощью микрокомпьютеров, и даже тех задач, которые раньше решались исключительно при помощи микроконтроллеров. Казалось бы, использование полноценного, хоть и мелкого компьютера для решения простых задач — это тот еще оверкилл, однако давайте разберемся, так ли это плохо? Эта статья является ответом на наш небольшой спор с хабровчанином devzona по этому поводу.

Предыстория

Казалось бы, что может быть более явной нишей для применения микроконтроллеров, чем автоматизация школьного звонка? Именно так думал неизвестный разработчик лет 5-7 назад, когда собирал вот такое замечательное устройство.



Собрано, судя по всему, на МК серии 8050, имеет на борту часики реального времени, умеет это время показывать на самодельной светодиодной матрице, и самое главное, умеет вовремя дергать релюшку, включающую школьный звонок. Устройство благополучно работает уже много лет, претензий к нему не было. Однако, все течет и меняется, и однажды простая Харьковская школа с углубленным изучением чего-то там решила пройти переаттестацию в лицей с еще более углубленным изучением того самого. Такая переаттестация, помимо всего прочего, требует перехода с 45-минутных уроков на пары, состоящие из двух академических часов по 40 минут. Тут-то и пришла беда. Разработчик часиков на МК благополучно ~~спился~~ уехал за границу, исходников не оставил, возможности перенастройки не предусмотрел. Именно с этой проблемой постучался ко мне в Скайп одним осенним днем мой друг Костя.

Осмотрев пациента пришло понимание, что быстрее чем за пару недель его переделать под требования заказчика не получится. По сути, нужно переписывать код с нуля. И, внезапно, вечером этого же дня курьер из DHL привез мне очередной Raspberry. Тут и пришла идея сделать свои часики, да не просто часики, а с магией. Ведь у нас есть целый микрокомпьютер с полноценным линуксом не

борту, руки развязаны, возможности безграничны!

Постановка задачи

Утром, после переговоров с заказчиком, задача была поставлена так: устройство должно конфигурироваться при помощи любого ПК, без дополнительного софта (дорого), уметь подтягивать точное время из интернета (по звонкам можно синхронизировать часы, все звонки строго с точностью до секунды), уметь работать автономно, и, как дополнительная опция на будущее, должны уметь получать конфигурацию звонков с удаленного сервера. Например, районо может самостоятельно выкладывать конфиг звонков для учебных заведений определенного типа. Задача поставлена, приступаем к реализации.

Для реализации проекта нам нужно следующее:

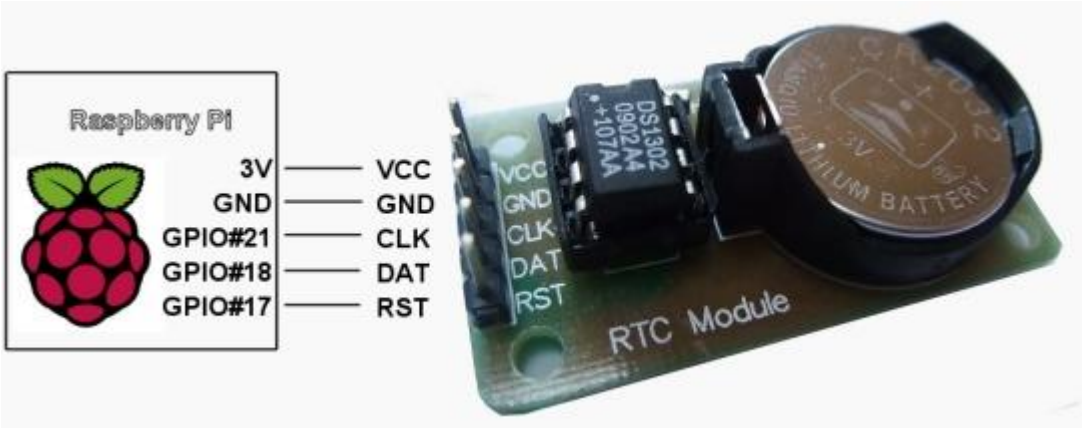
- Демон, умеющий дергать нужную GPIO ножку в нужное время
- Веб-интерфейс для конфигурирования времени звонков
- Часы реального времени
- Силовая электроника для управления школьными звонками

Я преднамеренно упускаю начальную конфигурацию Raspberry Pi, интернет полон материалами по установке дистрибутива, настройке сети, тайм-зоны и т.д.

Итак, приступим.

Часы реального времени

В качестве часов реального времени для устройства взял мелкую платку на DS1302, просто потому, что она нашлась у меня в кучке заказанного из Китая хлама. В сети обнаружилась замечательная статья, описывающая подключение именно этих часиков к малинке. Подключение довольно простое.



На этой же страничке доступен для скачивания софт, способный получать и записывать данные в эти RTC. Софт я немного переделал под себя, дабы визуализировать показания RTC перед синхронизацией с системным временем.

По правильному, часики должны обновляться в случае успешной синхронизации времени малинки с NTP сервером, и, если доступа к NTP серверу нет, тогда системные часы малинки должны быть синхронизированы с часами реального времени. Такой алгоритм необходим, так как DS1302 имеет привычку уползать на пару секунд в сутки, что неприятно. Однако, как заставить ntpd запускать скрипт после успешной синхронизации, я так и не нашел. Поэтому родился такой вот костыль:

/usr/local/bin/update_rtc

/etc/init.d/rtc

... и активируем автозагрузку:

```
sudo update-rc.d rtc defaults
```

Эти два файла позволяют синхронизировать системные часики малинки с RTC в случае, если после загрузки не обнаружена сеть, или обновить время в RTC, если сеть обнаружена. Через 30 сек после загрузки ntpd должен бы уже успеть обновить системные часы. В худшем случае, в RTC будет записано последнее время, когда Raspberry был включен. Я знаю, что это решение далеко не идеальное, но лучшего придумать не смог. Единственное, что приходит в голову — добавить строчку в крон для обновления RTC раз в 2-3 часа, дабы быть уверенным, что в часах реального времени более-менее точные данные. Если многоуважаемое сообщество подскажет лучшее решение — буду только рад.

Веб-сервер

Тут долго думать не пришлось. Основная задача сервера — показывать две странички и обрабатывать один POST запрос. Хрестоматийная реализация веб-сервера на Python просто напрашивается сама собой.

webserver.py

Из скуки и для лучшей расширяемости был даже добавлен простенький шаблонизатор. Обратите внимание, интерпретатор прописан в начале скрипта, так что после установки прав на исполнение, скрипт может быть запущен прямо из командной строки.

Что делает этот скрипт, думаю, понятно и без комментариев. Обработчик GET-запросов попросту отдает клиенту две формочки и главную страницу, заполняя переменную данными про текущее расписание. Обработчик POST-запросов сохраняет данные из формы в JSON-файл, который и является базой звонков.

Собственно, управлятор школьным звонком

Благодаря замечательной библиотеке GPIO для Python,
моргать ~~светодиодиком~~ школьным звонком с малинки очень просто. Этим занимается такой скрипт:

daemon.py

Скрипт создает новый поток, в котором проверяет время каждую секунду. Если время найдено в файле расписания, то на 5 секунд включаем звонок (подаем высокий уровень на ножку 25 GPIO). После каждого звонка перечитываем расписание, на случай, если оно было изменено из веб-интерфейса. Все прозрачно и просто.

Демонизируем и дрессируем смотрового пса

Действуя по аналогии с автозапуском синхронизации RTC, создаем следующие файлики:

/etc/init.d/schedule_daemon
/etc/init.d/schedule_webserver

И скрипты «сторожевых собачек» для них. Эти скрипты проверяют, запущен ли сервис, и, при необходимости, запускают его.

/etc/init.d/schedule_daemon_wd
/etc/init.d/schedule_webserver_wd

Аналогично, делаем эти скрипты автоматически загружаемыми при старте системы:

```
sudo update-rc.d schedule_daemon_wd defaults
```

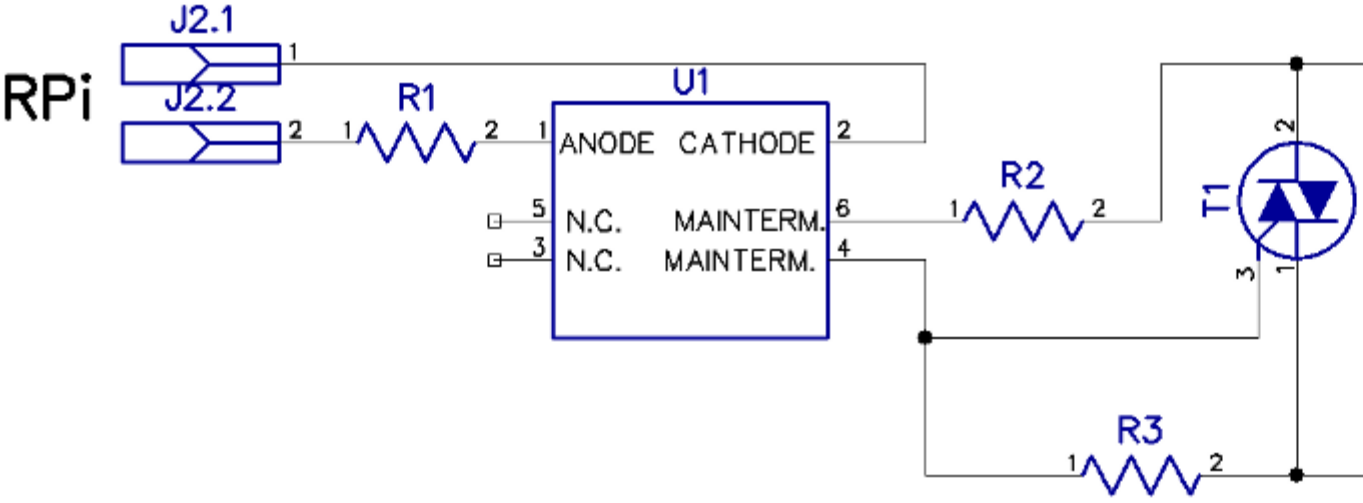
```
sudo update-rc.d schedule_webserver_wd defaults
```

И добавляем в крон новые задания:

/etc/cron.d/wd.cron
Теперь мы можем быть уверены, что оба демона запустились и будут стабильно работать. Не забываем добавить новую строчку в конце wd.cron, иначе crond будет его игнорировать!

Немного про силовую электронику

Вся силовая часть собрана совершенно стандартно. Суммарная мощность звонков в школе около 0.5 КВт, так что симистора BC137X в паре с оптроном MOC3061 вполне достаточно для коммутации этого хозяйства. Как показала практика, 3.3 вольта логической единицы достаточно для уверенного включения оптрона.



Можно было бы применить тут и реле, но как-то я не доверяю контактам, когда есть такие замечательные полупроводники. Фотографию макета преднамеренно не

выкладываю, т.к. до красивого монтажа так и не дошло.

Чего не хватает

Конечно, имея полноценный Linux-компьютер в своем распоряжении, можно «наворачивать» функциональность до бесконечности, причем времени на разработку уйдет сравнительно мало. Именно это обстоятельство говорит в пользу применения микрокомпьютеров для решения задач, с которыми, казалось бы, справится и микроконтроллер. Однако, все-же перечислю то, чего, по моему мнению, не хватает текущей реализации:

Во первых, безопасность. Стоило бы заморочиться на простой HTTP-Auth хотя бы, или, дописав немного скрипт, сделать базу паролей для входа в «админ-панель» системы. Да и над фильтрацией данных поработать стоит, как до, так и после отправки формы.
Во вторых, нужно бы добавить добавление/удаление академ. часов в форму. Внимательный читатель заметил, что это можно сделать попросту дорисовав в форму на клиентской стороне необходимые поля при помощи, например, простенького JavaScript кода.
В третьих, мне так хотелось сделать «тревожную кнопку» на главной, которая запускала бы звонок за 5-10 секунд. Пусть это будет маленькая задачка для пытливых умов читателей, благо, все необходимое для этого есть в статье.
В четвертых, не хватает блока бесперебойного питания. Ввиду отказа заказчика от разработки, до него мы так и не дошли.

Чем всё закончилось

К сожалению, Харьковская гимназия с углубленным изучением чего-то там решила, что собрать по 3 гривны с каждого родителя это очень, очень трудно, и нам в итоге дали от ворот поворот, поэтому реализация остановилась на действующем прототипе, который не содержит некоторых важных для конечной системы элементов. Но время, потраченное на разработку не прошло даром. Опыт разработки приложений для работы с железом на Python мне, надеюсь, не раз пригодится в жизни, тем более на загородном участке заканчивается строительство дома, в котором предусмотрена возможность управления всем из единого мозгового центра. Если смог управлять звонком, то и лампочки по расписанию включать смогу.

Послесловие

Надеюсь, уважаемые читатели, мне удалось донести до вас главную мысль.

Применение микрокомпьютеров для тривиальных, казалось бы, задач может поднять их реализацию на принципиально новый уровень, и вместо простейших реализаций, проприетарных протоколов и сложной поддержки, мы получаем гибкую систему с практически бесконечной расширяемостью, что в будущем выльется не только в удобство использования, но и в существенную экономию средств.

На реализацию всего вышеописанного было потрачено чуть более трех часов. Доведение до ума того, что есть требует еще столько же. Традиционно попрошу не пинать сильно за кривоватый местами код и возможные ошибки. Это моя первая статья на Хабре, и первый реализованный проект на Python. Всегда рад поправкам, пожеланиям и предложениям. Скриншоты и видео работы выложу по требованию.