

Documentation of the Arduino OpenPAYGO Implementation

Table of Contents

General	1
Adapt the codes to your need	1
Limitations	2
Recommendations	2
Main features	2
Factory setup	2
Token entry	2
Other features	3
Arduino disconnected from power	3
Memory	3
EEPROM	3
NVRAM (only if RTC module)	4
Modules	4
IR remote controller	4
Membrane keypad	4

General

Adapt the codes to your need

- Choose the interface by uncommenting one of the following lines in the code:
 - o `//#define KEYBOARD_MODE`
 - o `//#define IR_RECEIVER_MODE`
 - o `//#define MEMBRANE_KEYPAD_MODE`
 - o `//#define SERIAL_MODE`
- Choose the time management mode by uncommenting one of the following lines in the code:
 - o `//#define SET_TIME`
 - o `//#define ADD_TIME`

- Choose the time manager by uncommenting one of the following lines in the code
 - o `// #define ARDUINO_TIME_MANAGER`
 - o `// #define RTC_MODULE_TIME_MANAGER`
- If you are updating the code, uncomment `#define DEBUG_MODE` to open serial communication. Otherwise it won't open once the factory setup was done, to avoid hacking through UART
- If you want to modify the time divider, modify the line `#define TIME_DIVIDER 1`

Limitations

- Only works with 9 digits length code, not 12 or 15
- Restricted Digit Set Mode was not implemented
- Counter Synchronisation feature was not implemented
- A timestamp is regularly stored in the EEPROM to make sure the Arduino is not disconnected, but because EEPROM can only be updated around 100 000 times (roughly), the timestamp is only stored every hour (corresponds to 43 800 times for a 5 years lease). Update the line `#define EEPROM_TIME_UPDATE_PERIOD 3600` if you wish to modify it
- With the RTC module (if you use one), if its power lines are disconnected from the Arduino after initialization (in the factory or by a user), it will completely mess up the time management. This could be solved by storing timeStamps in the NVRAM and setting a difference between this timestamp and now from which we consider it was disconnected
- Specific scenario: with `ADD_TIME`, if a client adds 30 days, then disables PAYG, wait for 5 days, then enables PAYG with 1 day, he will have 26 days, not 1

Recommendations

- Recommended to initialize the EEPROM before the factory setup (put all the data to 0, by default all pins are usually HIGH)
- If an RTC module is used, all the bits of its NVRAM should be put to low (0)
- When the Arduino is flashed, make sure the UART pins are not connected, it usually raises an error and prevents the flashing

Main features

Factory setup

- If openPAYG controller receives all the data from the factory Controller (Serial Number, Starting Code, and Secret Key), it blinks three times and put SerialCount to 1 in the EEPROM

Token entry





- Start every token by `*`
- A code won't be considered until 9 keys are entered
- If a key is wrongly pressed during the token entry, just press `*` again and start again from scratch
- `MAX_TIME_TOKEN_ENTRY` sets the maximum time in seconds to type the token (20sec by default). If this time is exceeded, it aborts the token entry. This time should be smaller than `MAX_TIME_WITHOUT_UPDATE`, otherwise, token entry could

be interpreted as power disconnections if they occur just before the time stamp is supposed to be stored

- Token answers:
 - o If the token is wrong => blinks 10 times
 - o If the token disables PAYG mode => blinks 5 times, update the last count and paygDisabled in EEPROM
 - o If the token is correct => blinks 2 times (whatever time is added), update the last count in EEPROM
- To enable PAYG mode again, just use a functioning token with the number of days you want to add (you can add 0)

Other features

Arduino disconnected from power

- With ARDUINO_TIME_MANAGER: a disconnection (Arduino – power) is spotted if the lastTimeStamp stored in the EEPROM is superior to 0, during the setup of Arduino:
 - o  User: his activation goes to 0, he cannot use the machine until he gets another code
 - o  Machine: Arduino spots a difference between the lastTimeStamp in EEPROM and the new time it receives (close to 0), it puts activeUntil to 0 and store the new time in the EEPROM, nbDisconnections is incremented in the EEPROM
- With RTC_MODULE_TIME_MANAGER: a disconnection (Arduino – power) is spotted by checking if the timeInitializationRtc stored in the NVRAM is different from 0 during the setup of Arduino
 - o  User: nothing happens, he keeps the same activation
 - o  Machine: the Arduino spots that timeInitializationRtc is different from 0 and concludes that it was initialized before, therefore, a disconnection occurred: activeUntil and timeInitializationRtc are fetched in NVRAM of the RTC module, nbDisconnections is incremented in the EEPROM

Memory

EEPROM

The following data is stored in the EEPROM:

- Serial count (uint8_t): switches from 0 to 1 once the serial number, starting code and secret key are added (factory setup)
- Serial Number (uint32_t): added in factory setup
- Starting code (uint32_t): added in factory setup
- Secret key (unsigned char secretKey[16]): added in factory setup
- Last count (uint16_t): updated each time a correct activation was processed
- Last time stamp (uint32_t): updated every hour by default (can be modified by updating the line '#define EEPROM_TIME_UPDATE_PERIOD 3600')
- PAYG disabled (uint8_t): switches to 1 when PAYG is disabled, and back to 0 when PAYG is enabled again (by typing a new code that is not 998)
- Nb disconnections (uint8_t): incremented each time a disconnection is spotted

NVRAM (only if RTC module)

The following data is stored in the NVRAM of the RTC module:

- ActiveUntil (uint32_t): is updated every time a correct code is entered
- Initialized time (uint32_t): unix time when the NVRAM of the RTC module is initialized (could be in the factory or during the user's installation). This is not indispensable but much easier for debugging as it avoids going through Unix time all the time

Modules

IR remote controller

- Depending the IR remote controller used, modify the cases in the function 'getKeyPressed()' in 'f_TOKEN_IR'

Membrane keypad

- Depending the Keypad membrane used, modify the two lines (check the specs of the membrane you use)
 - o 'byte rowPins[ROWS] = {9, 8, 7, 6}; //connect to the row pinouts of the keypad'
 - o 'byte colPins[COLS] = {5, 4, 3}; //connect to the column pinouts of the keypad'