Micropython

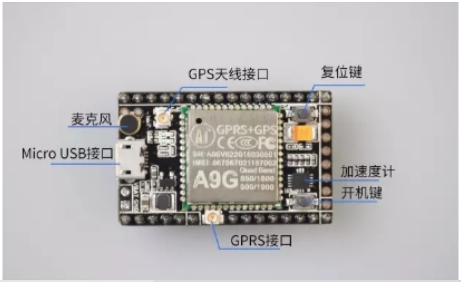
"MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments."

- Free and open-source
- Community-driven ($\sim$ 50 active users/month)
- Python 3.4, plans for 3.5
- Only 16k of RAM, 256k of ROM
- Wide support: win, linux, STM, ESP, javascript, ...
- Documented and tested

micropython.org

A9G

- 32 bit RISC core, frequency up to **312**MHz, with 4k instruction cache, 4k data cache
- 2G GSM (GPRS, calls, SMS) 800, 900, 1800, 1900 MHz
- **4Mb** RAM, **4Mb** ROM
- 29x GPIO
- 2x analog inputs (10 bit)
- 3x UART, 2x SPI, 3x I2C
- SDMMC, mic+audio, PMU (battery/USB) 3.8-5V
- optional GPS (separate chip via UART)
- from 5$



File: main.c

```
void main_micropy_thread(...) {
    ...
soft_reset:
    mp_stack_ctrl_init();  // Enable graceful OOM
    mp_stack_set_top(...);
    mp_stack_set_limit(...);
    gc_init(...); // Enable gc
    mp_init(); // Initialize mp
    ... optional: run startup scripts
    pyexec_event_repl_init();  // Event-based REPL
```

File: main.c

```c
    // Main loop for mp
    while (1) if (...) {
        while (Buffer_Gets(&fifoBuffer, &c, 1))
            if (pyexec_event_repl_process_char(c) {
                reset = 1;
                break;
            }
        if (reset) break;
    }
    gc_sweep_all();  // Release files, sockets
    mp_deinit();  // Bye, mp
    ... free other resources (heap, ...)
    goto soft_reset;
}
```

File: modgps.c

```
STATIC mp_obj_t get_firmware_version(void) {
    char buffer[300];
    if (!GPS_GetVersion(buffer, 300)) {
        mp_raise_GPSError("...");
        return mp_const_none;
    }
    return mp_obj_new_str(buffer, strlen(buffer));
}
```

File: modgps.c

```
STATIC MP_DEFINE_CONST_FUN_OBJ_0(
    get_firmware_version_obj,
    get_firmware_version
);
STATIC const mp_map_elem_t gps_globals_table[] = {
    { MP_OBJ_NEW_QSTR(MP_QSTR___name__),
        MP_OBJ_NEW_QSTR(MP_QSTR_gps) },
    { MP_OBJ_NEW_QSTR(MP_QSTR_get_firmware_version),
        (mp_obj_t)&get_firmware_version_obj },
    ...
};
STATIC MP_DEFINE_CONST_DICT(
    gps_globals,
    gps_globals_table
);
const mp_obj_module_t gps_module = {
    .base = { &mp_type_module },
    .globals = (mp_obj_dict_t*)&gps_globals,
};
```

TODO

1. Port the rest of the exposed API
2. Automated on-board testing (software+hardware testing, etc.)
3. Clean up the build process and main.c
4. Explore blobs for undocumented low-level API and port it
5. PR to mp repo

Thank you!

Port page: https://github.com/pulkin/micropython
Support: bug reports, PR, donations: see the link above
Support micropython: https://store.micropython.org/