Makey

- Chip: **ATmega328P**
- Addr: **0x51**

```
# init --- Effective IO: PORTA | PORTC
unit_makey = unit.get(unit.MAKEY, unit.PORTA)
# data:-1, 0 ~ 15
data = unit_makey.value
```

IO/ADC/DAC

PIR

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_pir = unit.get(unit.PIR, unit.PORTB)
# return 0 or 1
state = pir.state
```

Button

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_btn = unit.get(unit.BUTTON, unit.PORTB)

# Method
# if set the callback param, it will interrupt callback function
# or if not set param it will return result(True or False) at once
unit_btn.wasPressed(callback=None)
unit_btn.wasReleased(callback=None)
unit_btn.wasDoublePress(callback=None)
# holdTime: sec
unit_btn.pressFor(holdTime=1.0, callback=None):

#example
def on_wasPressed():
  lcd.print('Button was Pressed/n')

def on_wasReleased():
  lcd.print('Button was Released/n')

def on_pressFor():
  lcd.print('Button press for 1.2s press hold/n')

def on_doublePress():
  lcd.print('Button was double press/n')

unit_btn.wasPressed(on_wasPressed)
unit_btn.wasReleased(on_wasReleased)
unit_btn.wasDoublePress(on_doublePress)
unit_btn.pressFor(1.2, on_pressFor)
```

Dual Button

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_dualButton = unit.get(unit.DUAL_BUTTON, unit.PORTB)
# btnRed, btnRed method like button unit
btnRed = unit_dualButton.btnRed
btnRed = unit_dualButton.btnBlue
```

Relay

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_relay = unit.get(unit.RELAY, unit.PORTB)
# select com connect on
unit_relay.on()
# select com connect off
unit_relay.off()
```

IR

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_ir = unit.get(unit.IR, unit.PORTB)
# ir tx 25hz signal with 38khz signal carrier
unit_ir.txOn()
# off tx off
unit_ir.txOff()
# if receive ir signal return 1 else return 0
value = unit.rxStatus()
```

Angle

```
# init --- Effective IO: PORTB
unit_angle = unit.get(unit.ANGLE, unit.PORTB)

# get raw value, range 0 ~ 4095
adValue = unit_angle.readraw()

# get filter value, range 0 ~ 1023
filterValue = unit_angle.read()
```

Light

```
# init --- Effective IO: PORTB
unit_light = unit.get(unit.LIGHT, unit.PORTB)
# adc value, range 0 ~ 1024
analogValue = unit_light.analogValue
# digital value, only 0 or 1, base rotate the knob
digitalValue = unit_light.digitalValue
```

Earth

```
# init --- Effective IO: PORTB
```

```
unit_earth = unit.get(unit.LIGHT, unit.PORTB)
# adc value, range 0 ~ 1024
analogValue = unit_earth.analogValue
# digital value, only 0 or 1, base rotate the knob
digitalValue = unit_earth.digitalValue
```

Neopixel

- Chip: **SK6812**

```
# init --- Effective IO: PORTA | PORTB | PORTC, number --> 0 ~ 1023
unit_Neopixel = unit.get(unit.NEOPIXEL, unit.PORTB, number)
# pos: 1 ~ number
# color: 0x000000 ~ 0xffffff rgb888
unit_Neopixel.setColor(pos, color)
# 0 < posBegin < posEnd < number
# color: 0x000000 ~ 0xffffff rgb888
unit_Neopixel.setColor(posBegin, posEnd, color)
# color: 0x000000 ~ 0xffffff rgb888
unit_Neopixel.setColorAll(color)
# brightness: 0~255
unit_Neopixel.setBrightness(brightness)
```

RGB

- Chip: **SK6812**

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_Rgb = unit.get(unit.RGB, unit.PORTB)
# pos: 1 ~ number
# color: 0x000000 ~ 0xffffff rgb888
unit_Rgb.setColor(pos, color)
# 0 < posBegin < posEnd < number
# color: 0x000000 ~ 0xffffff rgb888
unit_Rgb.setColor(posBegin, posEnd, color)
# color: 0x000000 ~ 0xffffff rgb888
unit_Rgb.setColorAll(color)
# brightness: 0~255
unit_Rgb.setBrightness(brightness)
```

Weight

- Chip: **HX711**

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_weight = unit.get(unit.WEIGHT, unit.PORTA)
# 0 ~ 16777216 (24bit)
rawData = unit_weight.rawData
# get weight
weight = unit_weight.weight
# set weight to zero
unit_weight.zero()
```

Servo

```
# init --- Effective IO: PORTA | PORTB | PORTC
unit_servo = unit.get(unit.SERVO, unit.PORTA)
# us: 500 ~ 2500, 50HZ, High level duration
unit_servo.write_us(us)
# degrees: 0 ~ 180,  degree
unit_servo.write_angle(degrees)
```

UART

Finger

- Chip: **F

PC1020A**

```
# init --- Effective IO: PORTA | PORTC
unit_finger = unit.get(unit.FINGER, unit.PORTC)
# Method
# will change if finger state change
unit_finger.state

# user_id: 0 ~ 255
# access: 1, 2, 3
# note: if call this fun, unit_finger.state -> 'Wait add finger'
# finish: unit_finger.state -> 'Wait add finger'
# fail: unit_finger.state -> 'Add user fail'
unit_finger.addUser(user_id, access)

# if read know finger, will callback fingerCb with 2 formal parameter
def fingerCb(user_id, access):
  if user_id == 1 and access == 2:
    pass
unit_finger.readFingerCb(callback=fingerCb)

# user_id: 0~255
# finish: unit_finger.state -> 'Delete user finish'
# fail: unit_finger.state -> 'Delete user fail'
unit_finger.removeUser(user_id)

# remove all user data
unit_finger.removeAllUser()
```