|  |  |
| --- | --- |
| Name | Motor  '''A motor driver interface encapsulated in a Python class. Works with motor drivers using separate PWM and direction inputs such as the DRV8838 drivers present on the Romi chassis from Pololu.''' |
| Attributes | PWM\_pin: TimerChannel  '''Motor driver PWM TimerChannel object'''    DIR\_pin: Pin  '''Motor driver direction pin'''    nSLP\_pin: Pin  '''Motor driver NOT sleep (Enable) Pin''' |
| Methods | \_\_init\_\_(PWM: tuple[Timer, int, Pin], Dir: Pin, nSLP: Pin)  '''  Initializes a Motor object  PWM: Information required to set up PWM pin (Timer,channel\_num,Pin)  DIR: Direction Pin  nSLP: not Sleep Pin  '''  set\_effort(effort: int)  '''Sets the present effort requested from the motor based on an input value between -100 and 100'''    enable()  '''Enables the motor by taking it out of sleep mode into brake mode'''    disable()  '''Disables the motor driver by taking it into sleep mode''' |

|  |  |
| --- | --- |
| Name | Encoder  '''A quadrature encoder decoding interface encapsulated in a Python class''' |
| Attributes | ENC: Timer  '''Encoder Timer Object'''  ChA\_pin: TimerChannel  '''Encoder channel A timer channel in ENC\_AB mode'''  ChB\_pin: TimerChannel  '''Encoder channel B timer channel in ENC\_AB mode'''  position: int  '''Total accumulated position of the encoder'''  prev\_count: int  '''Timer Counter from the previous update'''  prev\_time: ticks  '''Ticks from most recent update'''  delta: int  '''Change in count between last two updates'''  dt: ticks  '''Amount of time between last two updates''' |
| Methods | \_\_init\_\_(self, tim: int, chA\_pin: Pin, chB\_pin: Pin):  '''  Initializes an Encoder object  tim: Timer number  chA\_pin: Channel A Pin  chB\_pin: Channel B Pin  '''  update()  '''Runs one update step on the encoder's timer counter to keep track of the change in count and check for counter reload'''  get\_position() -> int  '''Returns the most recently updated value of position as determined within the update() method'''    get\_velocity() -> float  '''Returns a measure of velocity using the most recently updated value of delta as determined within the update() method'''    zero()  '''Sets the present encoder position to zero''' |

|  |  |
| --- | --- |
| Name | LineSensor  '''Single channel IR line sensor interface encapsulated in a Python class''' |
| Attributes | adc: ADC  '''Sensor ADC Object'''    num: int  '''Number associated with sensor channel'''    white\_val: int  '''Calibrated value for white'''    black\_val: int  '''Calibrated value for black''' |
| Methods | \_\_init\_\_(self, adc\_Pin: Pin, num: int):  '''  Initializes a Line sensor object  adc\_Pin: Analog input Pin  num: number associated with sensor  '''  calibrate\_white()  '''Calibrate the sensor to what value represents white'''    calibrate\_black()  '''Calibrate the sensor to what value represents black'''    read() -> float  '''Read the sensor taking into account calibration and return float between 0 and 1''' |

|  |  |
| --- | --- |
| Name | SensorArray  '''Multi-channel IR Line Sensor interface''' |
| Attributes | CNTRL: Pin  '''Sensor dimming control pin'''    sensor\_list: list[Line\_Sensor]  '''List of line sensors in the array'''    dimm\_lvl: int  '''Current Dimming level of the array'''  sensor\_values: list[float]  '''Most recent read sensor values''' |
| Methods | \_\_init\_\_(self, CNTRL: Pin, sensor\_define: list[tuple[Pin, int]])  '''  Initializes a Line Sensor Array object  CNTRL: Sensor dimming control pin  sensor\_define: List of tuples (adc\_Pin, num)  '''  dimm()  '''reduce the dimming level of the sensors by 1, each pulse reduces the intensity by 3.33% from 100%\* reduces by one level each time'''  calibrate\_all(color: str)  '''Calibrate all sensors for the color specified color: Color to calibrate for ("white" or "black")'''  read\_all() -> list[float]  '''Read all of the sensors in the array and return their normalized values'''  get\_centroid() -> float  '''Read all of the sensors and calculate the centroid of the distribution''' |

|  |  |
| --- | --- |
| Name | PIDController  '''Implement PID Control for a given system''' |
| Attributes | KP: float  '''KD: Derivative Controller Gain'''    KI: float  '''KI: Integral Controller Gain'''    KD: float  '''KD: Derivative Controller Gain'''  prev\_err: Any  '''Previous error in consistent units'''    integral\_err: Any  '''Integral of error in consistent units'''  prev\_ticks: ticks  '''Previous time (us)''' |
| Methods | \_\_init\_\_(self, KP: float, KI=0., KD=0.):  '''  Initializes PID Controller Object  KP: Proportional Controller Gain      KI: Integral Controller Gain      KD: Derivative Controller Gain  '''  reset()  '''Reset error values for clean initial start-up'''  controller(setpoint, feedback) -> tuple  '''Run 1 loop of closed-loop feedback PIC controller with given setpoint  and feedback, assume consistent units''' |

|  |  |
| --- | --- |
| Name | BNO055  '''A BNO055 IMU interface encapsulated in a Python class''' |
| Attributes | controller: I2C  '''I2C controller object'''    addr: 0x29  '''I2C Peripheral Address for the IMU'''  OPR\_MODE: 0x3D  '''Memory address for Operation Mode Register'''  calibration: bytearray  ''' |
| Methods | \_\_init\_\_(controller: I2C)  '''  Initializes BNO055 object  controller: I2C object preconfigured in CONTROLLER mode  '''  \_write\_reg(reg, value: bytearray)  '''Write a value to a given memory register'''  \_read\_reg(reg) -> tuple  '''Read a specific memory register value'''  \_set\_mode(mode)  '''Change operating mode of IMU'''  get\_status() -> tuple[bool, bool, bool, bool]  '''Retrieve and parse the calibration status byte from the IMU'''  read\_calibration() -> bytearray  '''Retrieve the calibration coefficients from the IMU as a binary data and writes it to calibration.txt'''  write\_calibration() -> bytearray  '''write calibration coefficients back to the IMU from pre-recorded binary data in calibration.txt'''  get\_value() -> tuple  '''read value from the IMU'''  angle\_diff(angle1:int,angle2:int) -> float  '''Computes the angle difference angle1-angle2 such that the value is between -180 and 180''' |