CODE: ADIY FLY PICO with ADIY HLK B50 TTL Bluetooth Module

1. i2c_lcd.py

```
from lcd_api import LcdApi
from machine import I2C
from time import sleep_ms

# The PCF8574 has a jumper selectable address: 0x20 - 0x27
DEFAULT_I2C_ADDR = 0x27

# Defines shifts or masks for the various LCD line attached to the PCF8574

MASK_RS = 0x01
MASK_RW = 0x02
MASK_E = 0x04
SHIFT_BACKLIGHT = 3
SHIFT_DATA = 4


class I2cLcd(LcdApi):
    """Implements a HD44780 character LCD connected via PCF8574 on I2C."""

    def __init__(self, i2c, i2c_addr, num_lines, num_columns):
        self.i2c = i2c
        self.i2c_addr = i2c_addr
        self.i2c.writeto(self.i2c_addr, bytearray([0]))
        sleep_ms(20)   # Allow LCD time to powerup
        # Send reset 3 times
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        sleep_ms(5)    # need to delay at least 4.1 msec
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        sleep_ms(1)
        self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
        sleep_ms(1)
        # Put LCD into 4 bit mode
        self.hal_write_init_nibble(self.LCD_FUNCTION)
        sleep_ms(1)
        LcdApi.__init__(self, num_lines, num_columns)
        cmd = self.LCD_FUNCTION
        if num_lines > 1:
            cmd |= self.LCD_FUNCTION_2LINES
        self.hal_write_command(cmd)

    def hal_write_init_nibble(self, nibble):
        """Writes an initialization nibble to the LCD.
```

```
        This particular function is only used during initialization.
        """
        byte = ((nibble >> 4) & 0x0f) << SHIFT_DATA
        self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytearray([byte]))

    def hal_backlight_on(self):
        """Allows the hal layer to turn the backlight on."""
        self.i2c.writeto(self.i2c_addr, bytearray([1 << SHIFT_BACKLIGHT]))

    def hal_backlight_off(self):
        """Allows the hal layer to turn the backlight off."""
        self.i2c.writeto(self.i2c_addr, bytearray([0]))

    def hal_write_command(self, cmd):
        """Writes a command to the LCD.

        Data is latched on the falling edge of E.
        """
        byte = ((self.backlight << SHIFT_BACKLIGHT) | (((cmd >> 4) & 0x0f) <<
SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytearray([byte]))
        byte = ((self.backlight << SHIFT_BACKLIGHT) | ((cmd & 0x0f) <<
SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytearray([byte]))
        if cmd <= 3:
            # The home and clear commands require a worst case delay of 4.1
msec
            sleep_ms(5)

    def hal_write_data(self, data):
        """Write data to the LCD."""
        byte = (MASK_RS | (self.backlight << SHIFT_BACKLIGHT) | (((data >> 4)
& 0x0f) << SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytearray([byte]))
        byte = (MASK_RS | (self.backlight << SHIFT_BACKLIGHT) | ((data & 0x0f)
<< SHIFT_DATA))
        self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
        self.i2c.writeto(self.i2c_addr, bytearray([byte]))
```

2. DisplayDataViaBluetooth.py

```
from machine import UART, Pin, I2C
import utime
from i2c_lcd import I2cLcd

# Configure UART pins
uart = UART(0, baudrate=115200, tx=Pin(0), rx=Pin(1))
i2c = I2C(1, sda=Pin(2), scl=Pin(3), freq=400000)
lcd = I2cLcd(i2c, 0x27, 2, 16)

def display_on_lcd(data):
    lcd.clear()
    lcd.putstr(data)

# Main loop
while True:
    # Read data from Bluetooth
    if uart.any():
        received_data = uart.readline().decode().strip()
        print(received_data)
        # Display received data on LCD
        display_on_lcd(received_data)
```