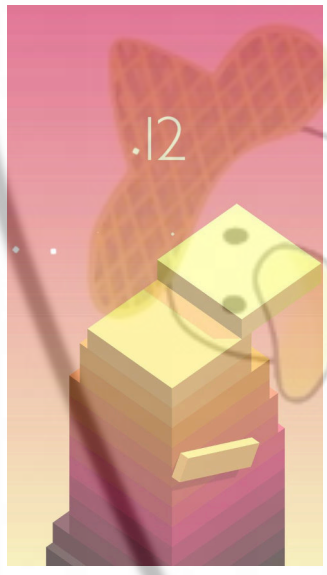
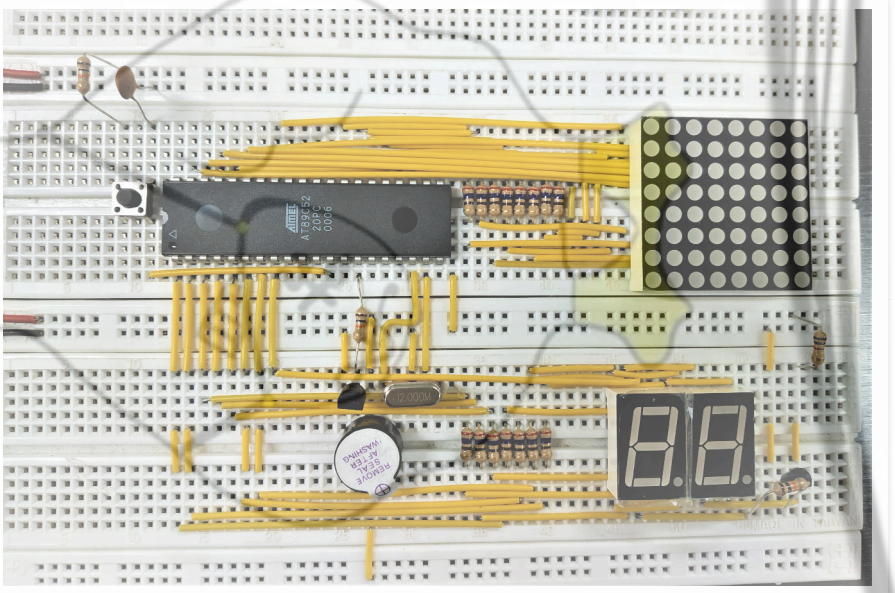
MICROCONTROLLER PRACTICE

8051 Stack Game

Inspired by a famous mobile game "Stack," where the player tries to stack moving blocks as high as possible, I and two other students made use of the knowledge we learned in our microcontroller practice course and built a circuit that mimics the game's functionality on a breadboard with an AT89S52.

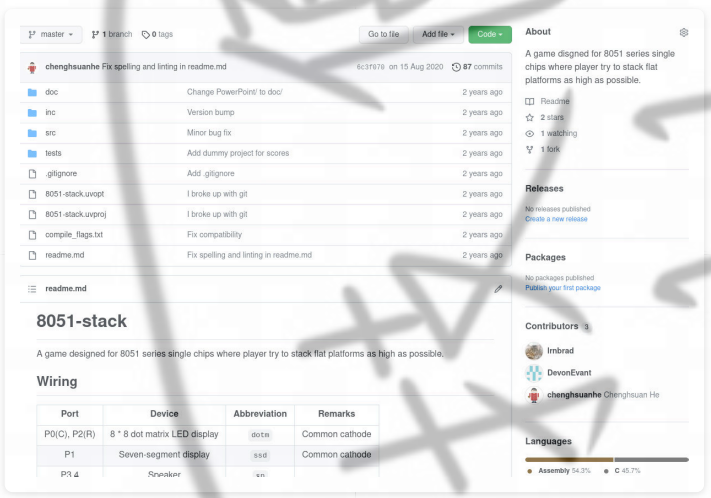


▲ Original game screenshot



▲ Breadboard circuit

We successfully replicated the game with a dot matrix display, a seven-segment display for the score, along with a buzzer for sound effects. During the designing of code and circuits, **I learned how to coordinate a team as a project leader.** I used Git to manage source code and documentation and **published them on GitHub with an open-source license.**



▲ Collaborate and open-source on GitHub

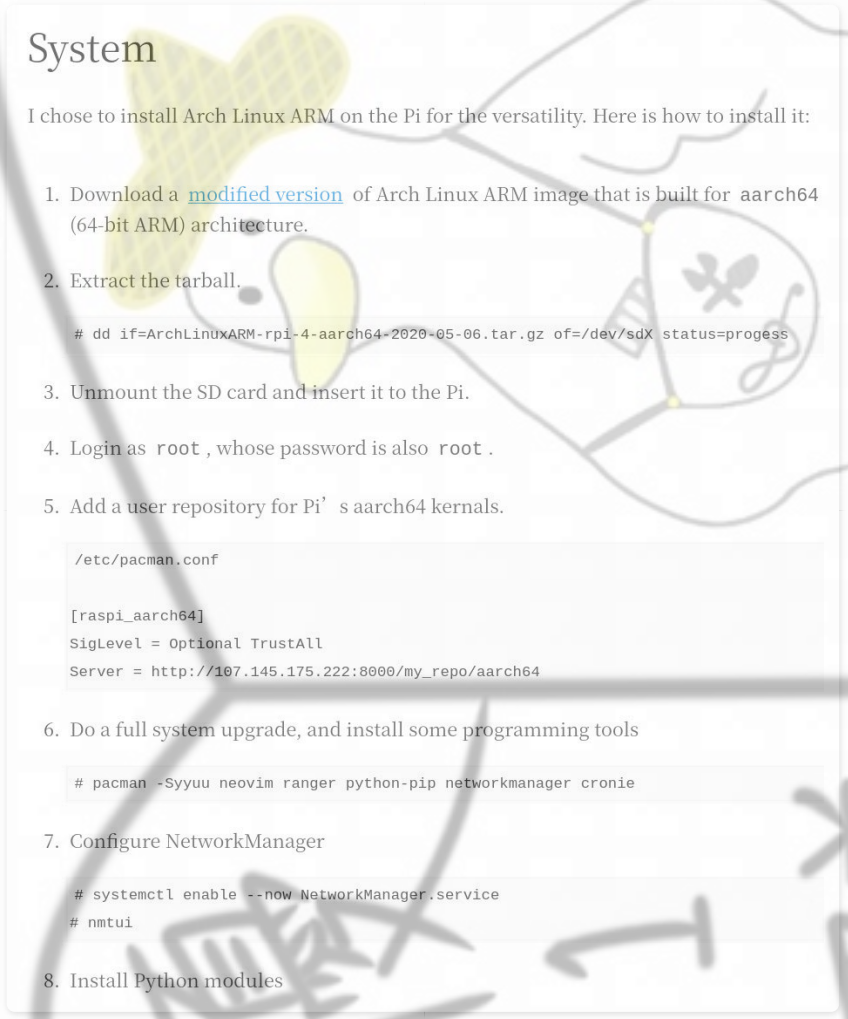
```
84 void onBtnPress() interrupt 0 {
85   u8 i;
86
87   // Align and cut off
88   for (i = 0; i < 7; i++) {
89     // If a LED is on but the below one isn't
90     if (dotm_buf[lineY] >> i & 1 > dotm_buf[lineY + 1] >> i & 1) {
91       dotm_buf[lineY] &= ~(1 << i); // that LED is cleared
92     }
93   }
94
95   // If all LEDs are cleared, game over
96   if (!dotm_buf[lineY]) {
97     gameOver = 1;
98     return;
99   }
100
101  // Move line upwards and autoscroll
102  lineY--;
103  if (lineY == 0) {
104    lineY = 1;
105    // Shift dotm_buf, 8 bytes long, insert 0, shift right
106    arr_shift(dotm_buf, 8, 0x00, 1);
107  }
```

▲ Well documented code

I/O INTERFACE CONTROL PRACTICE

Cloud Gas Leak Detector

Once upon a time, I was outside my Father's office, and I sensed a hard-to-breathe gas smell. I rushed to the back door and found out the cause was a leaking gas tube. Therefore to prevent further incidents alike, I decided to design a gas detector that notifies leakage via phones.



System

I chose to install Arch Linux ARM on the Pi for the versatility. Here is how to install it:

1. Download a modified version of Arch Linux ARM image that is built for `aarch64` (64-bit ARM) architecture.
2. Extract the tarball.

```
# dd if=ArchLinuxARM-rpi-4-aarch64-2020-05-06.tar.gz of=/dev/sdX status=progess
```

3. Unmount the SD card and insert it to the Pi.
4. Login as `root` , whose password is also `root` .
5. Add a user repository for Pi' s aarch64 kernals.

```
/etc/pacman.conf

[raspi_aarch64]
SigLevel = Optional TrustAll
Server = http://107.145.175.222:8000/my_repo/aarch64
```

6. Do a full system upgrade, and install some programming tools

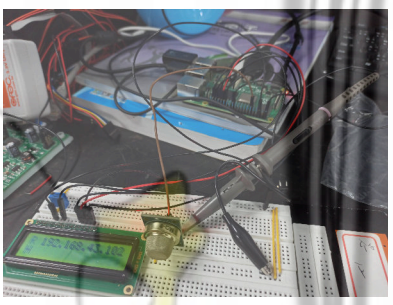
```
# pacman -Syyuu neovim ranger python-pip networkmanager cronie
```

7. Configure NetworkManager

```
# systemctl enable --now NetworkManager.service
# nmtui
```

8. Install Python modules

▲ Instructions for building the detector



▲ A working prototype.
IP address and sensor value is shown on the LCD

```
import RPi.GPIO as GPIO
from RPLCD import CharLCD
import time
import socket
import threading
from fbchat import Client
from fbchat.models import *
import sys

def onMQ9FallingEdge(channel):
    emergencyMessage = '🔥🔥🔥 WARNING: FLAMMABLE GAS CONCENTRATION TOO HIGH! 🔥🔥🔥'
    client.send(Message(text=emergencyMessage), thread_id=client.uid)
    client.send(Message(text=emergencyMessage), thread_id=recieveID)


class EchoBot(Client):
    def onMessage(self, author_id, message_object, thread_id, thread_type, **kwargs):
        self.markAsDelivered(thread_id, message_object.uid)
        self.markAsRead(thread_id)

        print("{} from {} in {}".format(
            message_object, thread_id, thread_type.name))

        # If you're not the author, echo
        if author_id != self.uid:
            self.send(Message(text='Time: {}\nMode: {}\nLocal IP address: {}\nMQ-9 Status: {}\nFacebook Messenger: {}'
                                   .format(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime()),
                                           'Debug' if debug else 'Normal',
                                           ip_addr(),
                                           'Safe' if GPIO.input(11) else 'Warn',
                                           'Connected' if chat_thread.is_alive() else 'Error')),
                      thread_id=thread_id,
                      thread_type=thread_type)
```

▲ Program that controls sensor and sends message

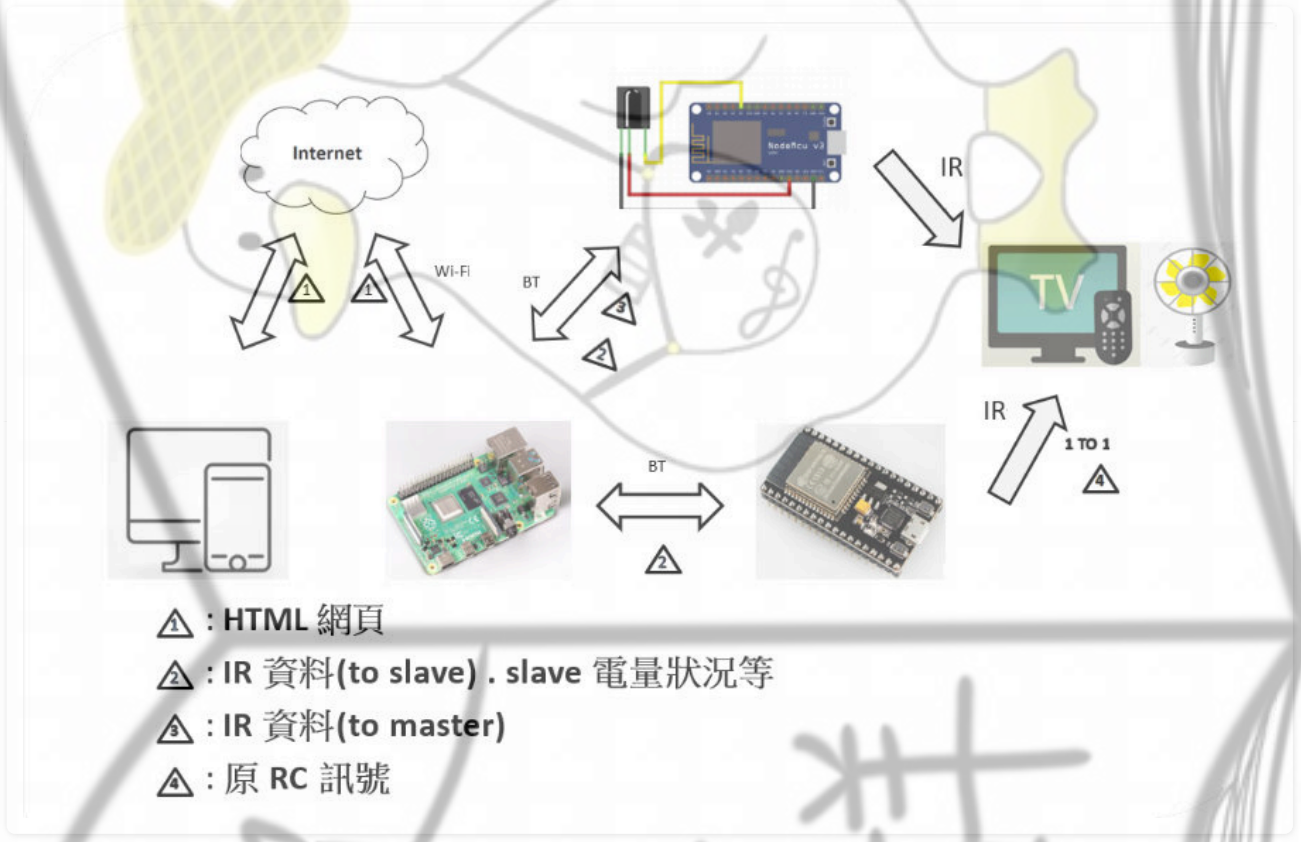
Powered by a Raspberry Pi, the detector displays network and sensor information on an LCD and sends a message to the user's phone via Facebook Messenger if it detects any flammable gas in the air.

Making the detector, **I learned how to control Raspberry Pi's GPIO** and send messages over the Internet using Python. **I created a step-by-step tutorial for one to build their own** in the hope of preventing more tragedies from occurring.

INDEPENDENT STUDY

Cross-Platform Remote Controller

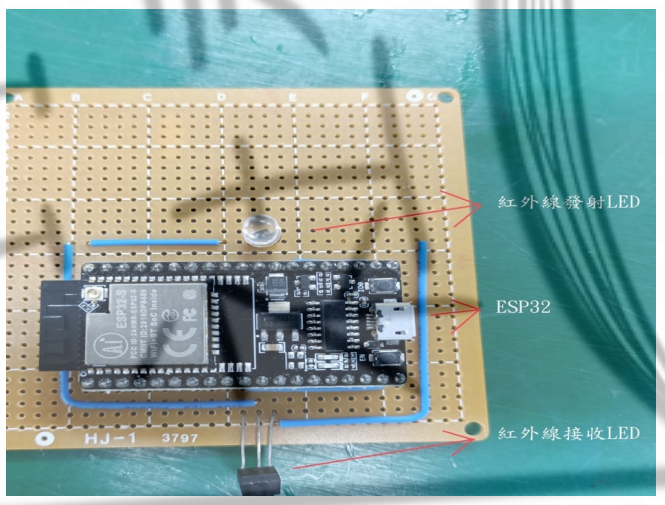
In an age where IoT devices are taking the appliance market by storm, controlling devices using a phone is becoming the norm. Older devices primarily use inferred remote controls as a means of controlling them. Keeping an IR remote for each device takes up valuable space, and replacing batteries is annoying and uneconomical. Therefore, **we made Cross-Platform Remote Controller (CPRC), a project attempting to smarten up old appliances in a modular, extensible manner.**



▲ Cluster diagram of the CPRC project

We chose the ESP32 microcontroller as our client for its low cost and low energy consumption. Via Bluetooth Low Energy, the microcontroller connects to a central computer (Raspberry Pi in this case) which provides a user interface on the web.

Using off-the-shelf components and standards-compliant format (JSON), **we designed a structure accessible for anyone to modify and extend.**



▲ a CPRC client prototype