PYCON KOREA 2017

아마존 에코를 활용한 음성 인식 에어컨 제어 A to Z

서주은 (zuneseo@buzzvil.com)

Back to the Basic

프로젝트 목표

- 사무실에 있는 에어컨을 쉽게 끄고 켤 수 있는 방법은 없을까?
  - 무려 3대의 에어컨이 존재
  - 심지어 모델마저 제각각이라 2개의 리모콘이 필요

프로젝트 목표

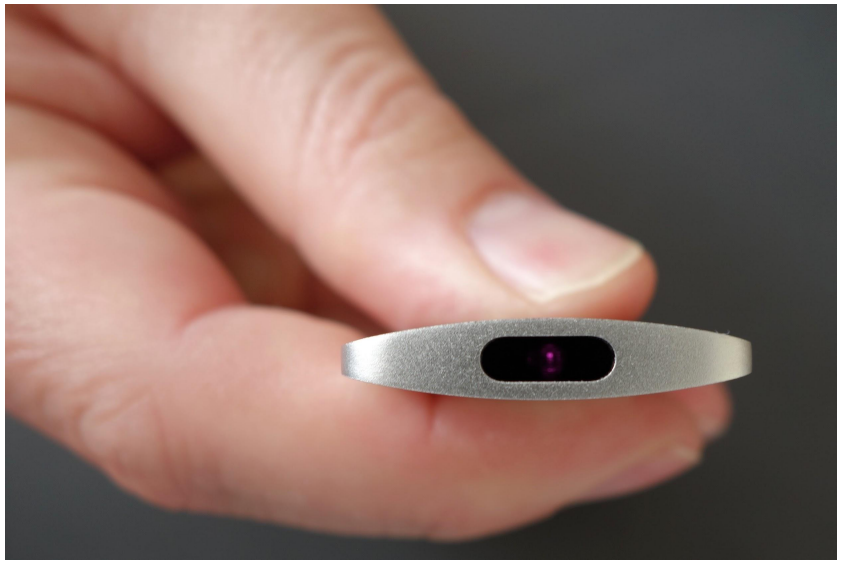
- 에코를 이용한 에어컨 제어
  - **Alexa, turn on the AC.**
  - **Alexa, turn off the air conditioner.**
  - **Alexa, could you please turn on the AC?**

프로젝트 목표

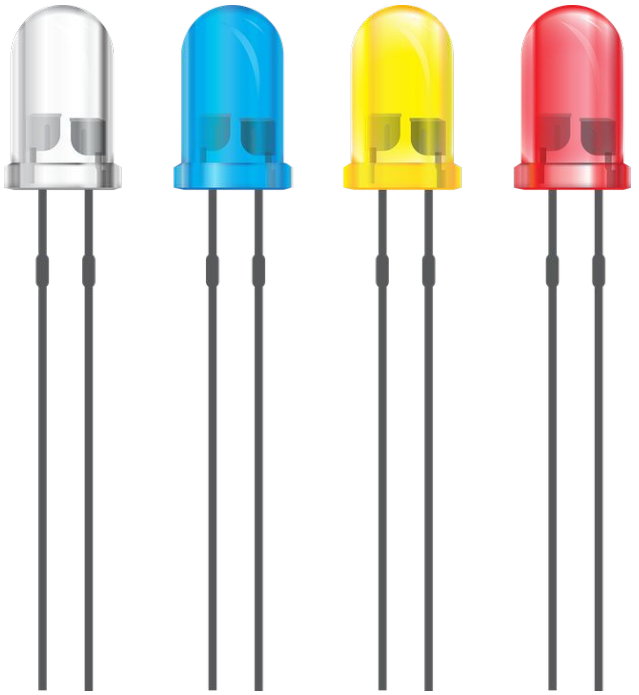
- **Slack 커맨드 만들기**
  - **/acon - 에어컨 끄기**
  - **/acoff - 에어컨 켜기**
  - **/acwarm - 에어컨 약하게**
  - **/acmedium - 에어컨 중간**
  - **/accool - 에어컨 세게**

하드웨어 만들기

- 리모컨
  - 적외선 통신
  - 단방향 통신
  - 에어컨의 상태는 리모컨에 저장되어 있다

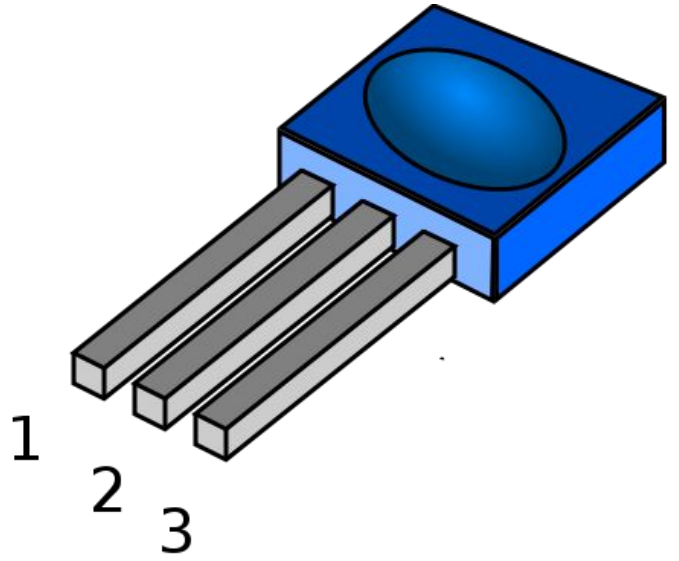


- 적외선 발신기
  - **LED**의 일종
  - 전기에너지를 빛 에너지로
  - 적외선/가시광선/자외선
  - 깜빡임의 주기를 통해 데이터를 전달



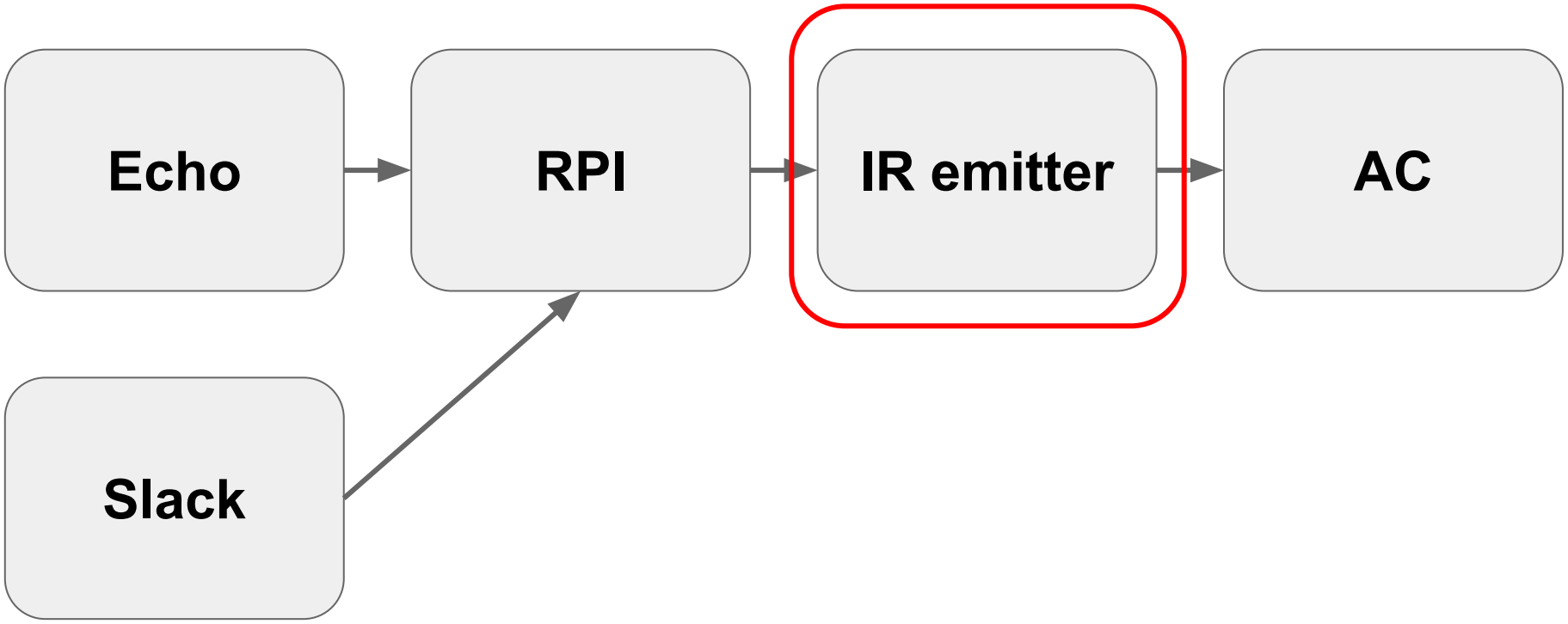
하드웨어 만들기

- 리모컨 프로토콜 알아내기
  - **IR receiver 이용해 캡쳐**

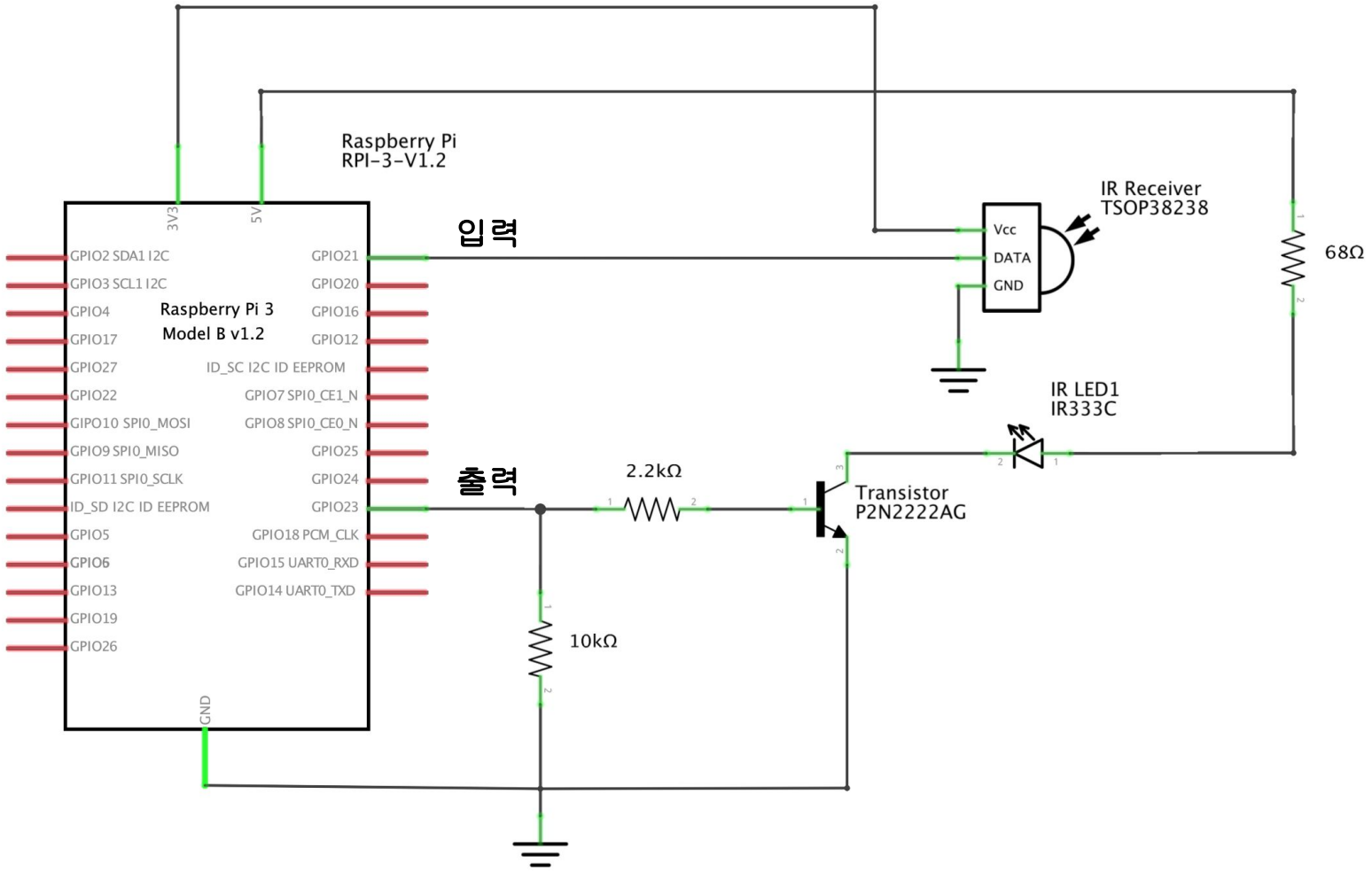


하드웨어 만들기

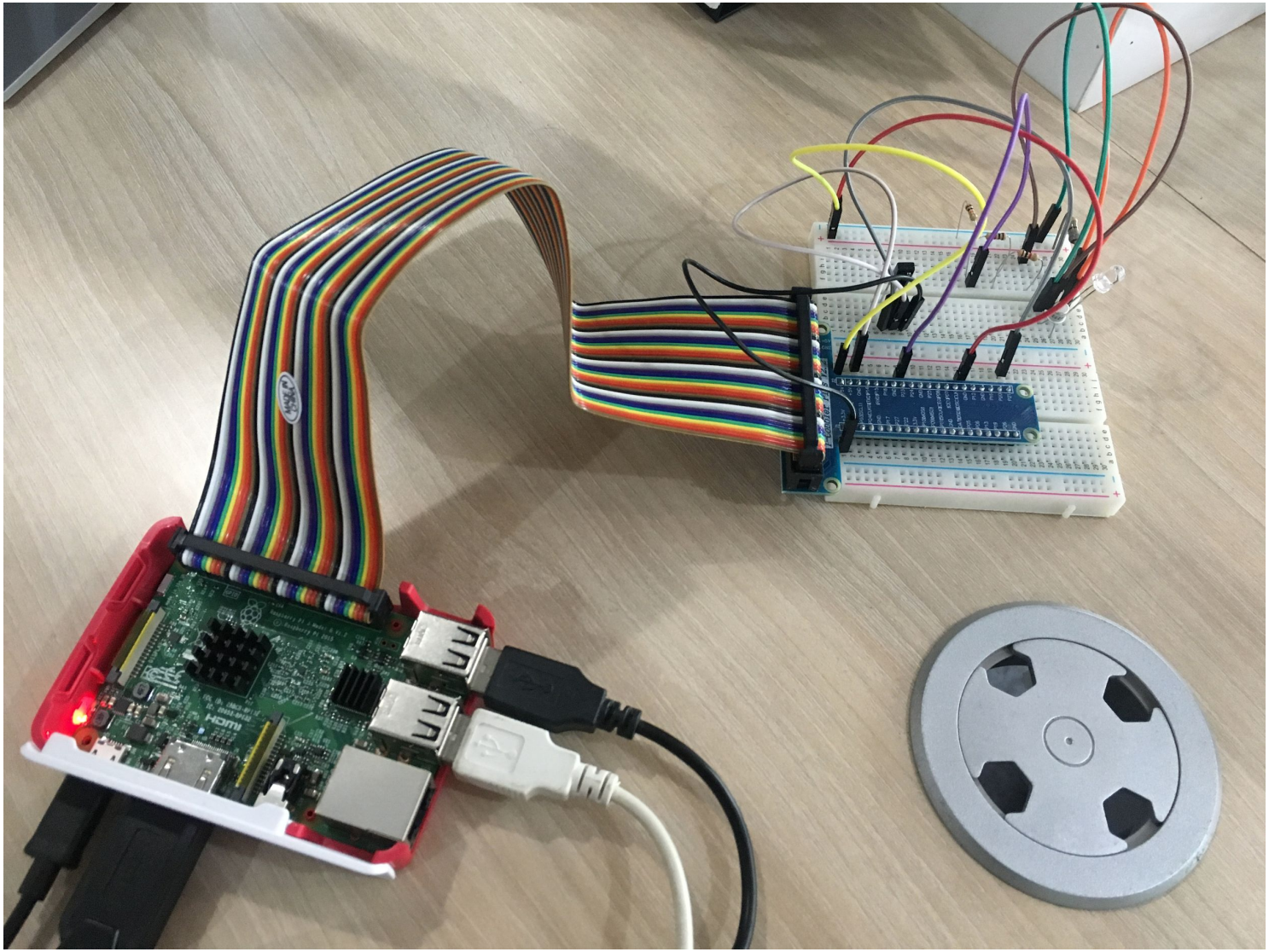
- Echo: Amazon Echo
- RPI: Raspberry pi
- AC: Air conditioner



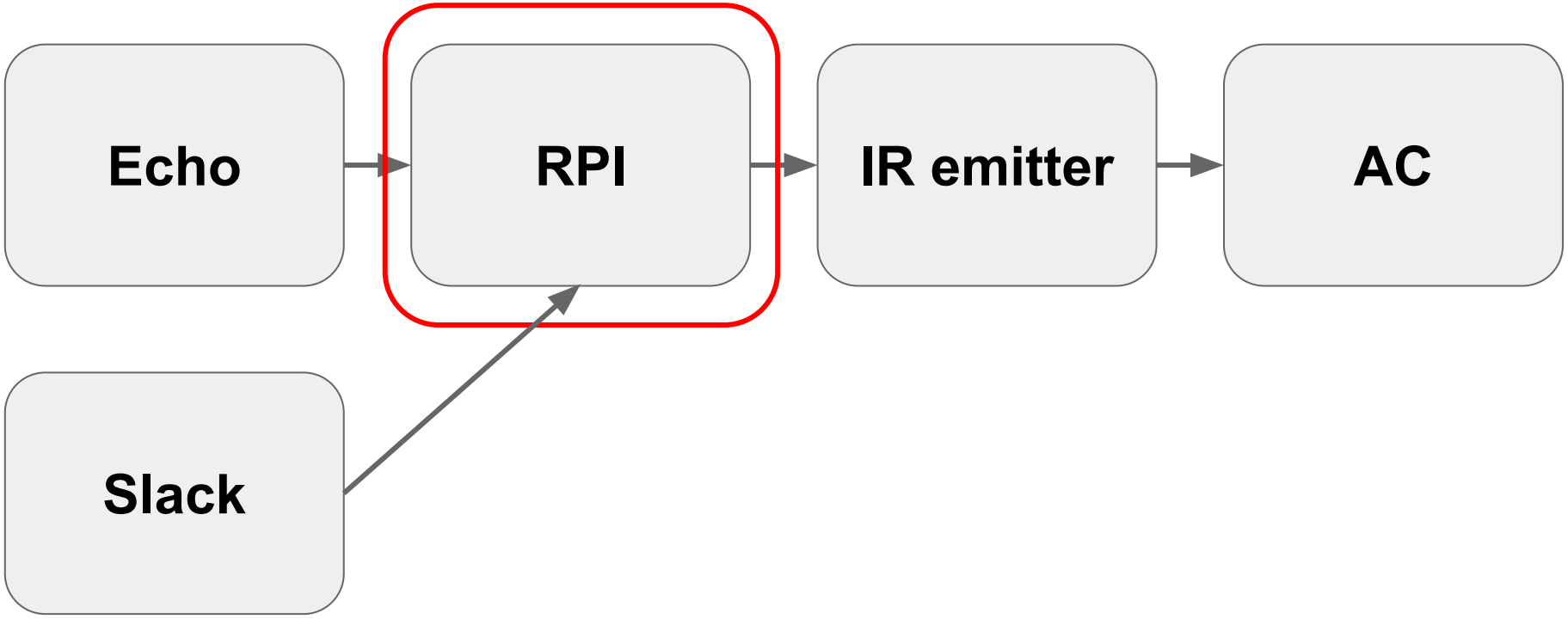
하드웨어 만들기



하드웨어 만들기



LIRC



LIRC

- **LIRC (Linux Infrared Remote Control)**
  - **적외선 신호를 보내고 받을 수 있게 해주는 패키지**
  - **sudo apt-get install lirc**
  - **LIRC commands**
    - mode2
    - irsend
    - irrecord
  - **LIRCD**
    - Client application이 Unix domain socket을 통해 쉽게 적외선 신호 송수신을 해주는 데몬

LIRC

- **irrecord**
  - irrecord -d /dev/lirc0 lircd.conf

LIRC

- **lircd.conf**

```
begin remote

  name  lg-ac
  bits           20
  flags SPACE_ENC|CONST_LENGTH
  eps            30
  aeps          100

  header       3204  9803
  one           575  1497
  zero          575   462
  ptrail        575
  pre_data_bits   8
  pre_data       0x88
  gap          106773
  toggle_bit_mask 0x0

      begin codes
          BTN_0                     0xC0051 0xFF312 # 끄기

          BTN_1                     0x00314 0xFF345 # 냉방/18도/1바람
          BTN_2                     0x00303 0xFF345 # 냉방/18도/2바람
          BTN_3                     0x00347 0xFF345 # 냉방/18도/4바람
      end codes

end remote
```

LIRC

- **LG 에어컨 프로토콜 분석**
  - 온도/바람 세기는 왼쪽 값에서 설정
  - 오른쪽 값은 누른 버튼에 대한 값
  - 단방향 통신의 제약으로 인해 에어컨의 모든 상태값을 항상 전송해야함
  - 전원 on 커맨드 = 전원 on + 온도 + 바람세기

```
begin codes
    BTN_0                    0xC0051 0xFF312 # 끄기

    BTN_1                    0x00314 0xFF345 # 냉방/18도/1바람
    BTN_2                    0x00303 0xFF345 # 냉방/18도/2바람
    BTN_3                    0x00347 0xFF345 # 냉방/18도/4바람

    BTN_4                    0x00617 0xFF345 # 냉방/21도/1바람
    BTN_5                    0x00606 0xFF345 # 냉방/21도/2바람
    BTN_6                    0x0064A 0xFF345 # 냉방/21도/4바람

    BTN_7                    0x0091A 0xFF345 # 냉방/24도/1바람
    BTN_8                    0x00909 0xFF345 # 냉방/24도/2바람
    BTN_9                    0x0094D 0xFF345 # 냉방/24도/4바람

    BTN_10                   0x00C1D 0xFF345 # 냉방/27도/1바람
    BTN_11                   0x00C0C 0xFF345 # 냉방/27도/2바람
    BTN_12                   0x00C50 0xFF345 # 냉방/27도/4바람

    BTN_50                   0x04B10 0xFF345 # 난방/26도/1바람
    BTN_51                   0x04B0F 0xFF345 # 난방/26도/2바람
    BTN_52                   0x04B43 0xFF345 # 난방/26도/4바람

    BTN_53                   0x04D12 0xFF345 # 난방/28도/1바람
    BTN_54                   0x04D01 0xFF345 # 난방/28도/2바람
    BTN_55                   0x04D45 0xFF345 # 난방/28도/4바람

    BTN_56                   0x04F14 0xFF345 # 난방/30도/1바람
    BTN_57                   0x04F03 0xFF345 # 난방/30도/2바람
    BTN_58                   0x04F47 0xFF345 # 난방/30도/4바람
end codes
```

LIRC

- 삼성 에어컨 프로토콜 분석
  - **irrecord**로 분석이 안됨
  - 체크섬 로직이 있다고 함
- **mode2** 사용
  - 저수준의 **IR waveform capture command**
  - 결과를 **lircd.conf** 포맷으로 출력해주는 **-m** 옵션 사용
  - sudo mode2 -m -d /dev/lirc0 > lirc.conf

LIRC

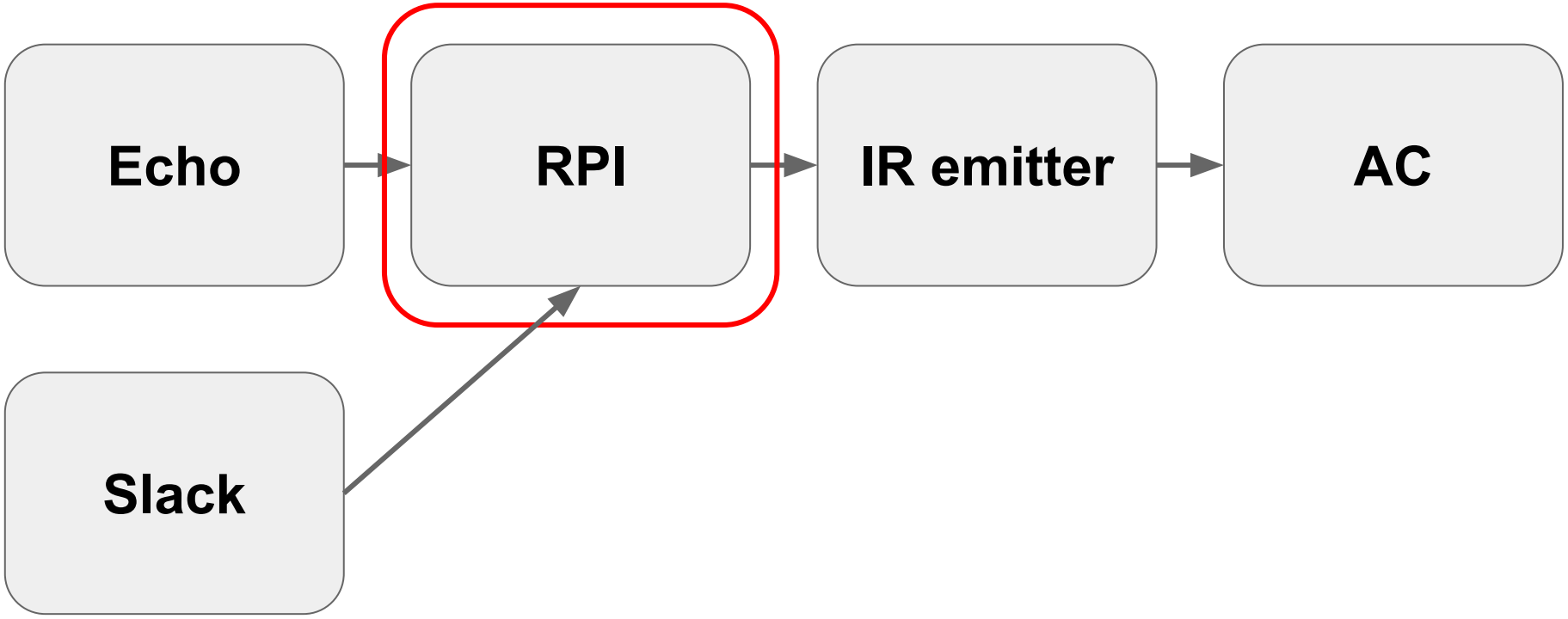
- 삼성.. A~C(..8) 노가다!

```
begin remote

  name  samsung-ac
  flags RAW_CODES
  eps            30
  aeps          100

  gap          8929

      begin raw_codes

          name BTN_0
               632    17684     3048     8887      541      444
               552     1440      545      441      556      436
               547      444      550      441      552      439
               554      439      556      435      550     1438
               551      440      554      436      557     1455
               535     1452      545      443      551     1435
```

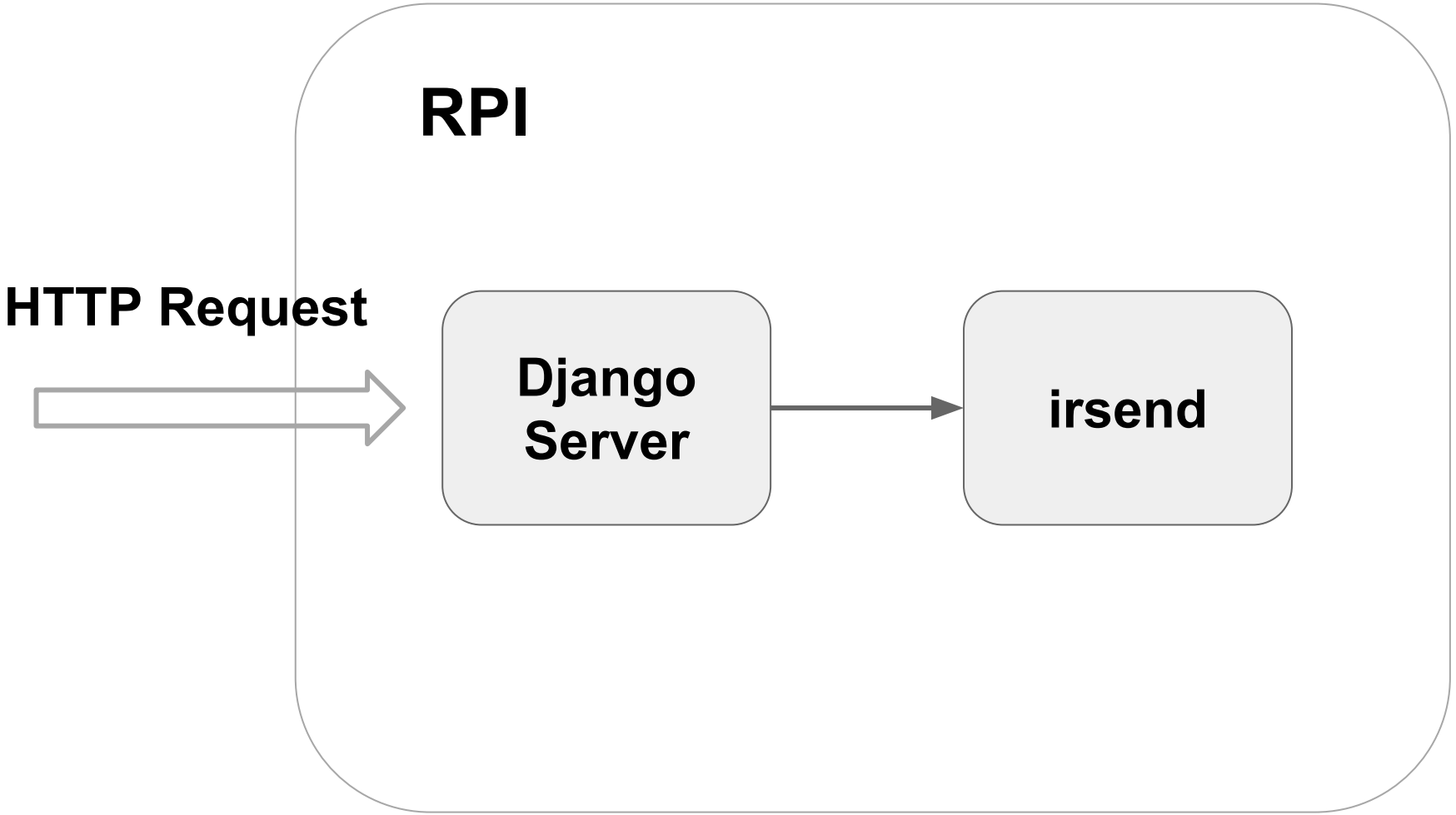
LIRC

- **irsend**
  - irsend SEND_ONCE lg-ac BTN_1
    - lg-ac 디바이스에 BTN_1에 해당하는 신호를 한 번 보내라
  - irsend SEND_ONCE samsung-ac BTN_2

LIRC



애플리케이션

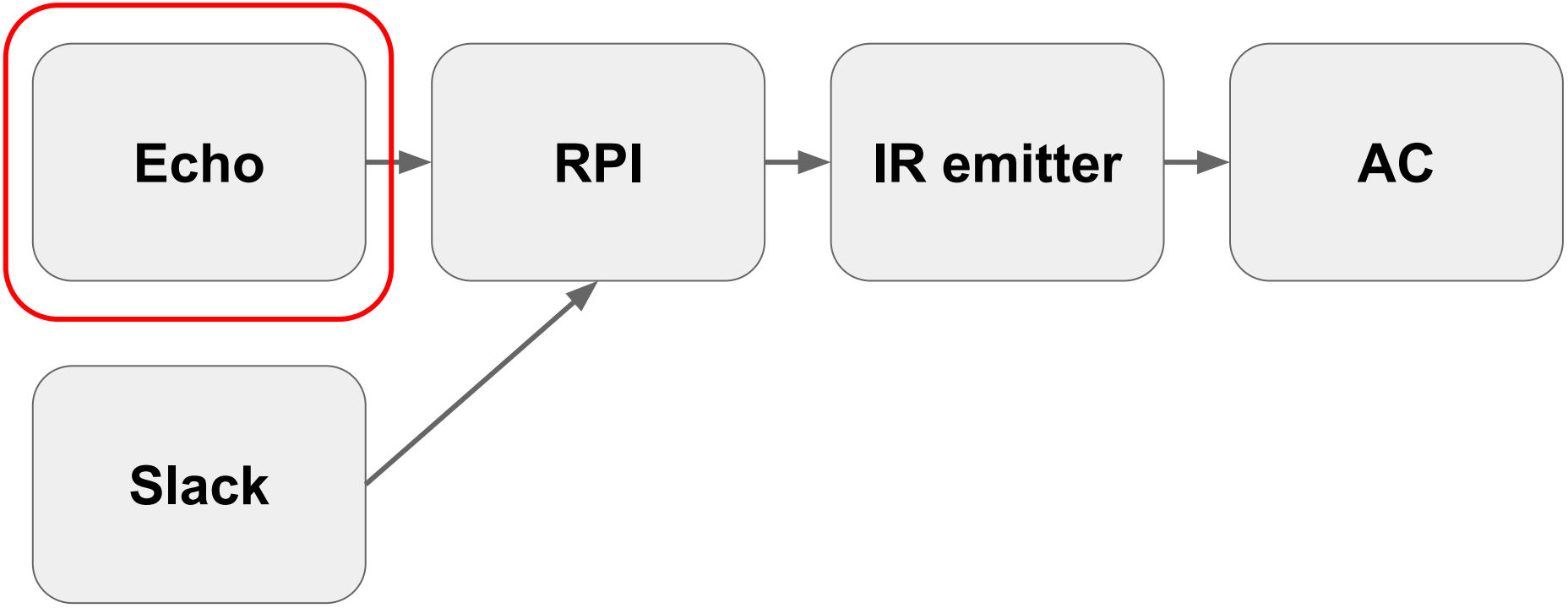


Django Server

http://127.0.0.1/api/ac/on/

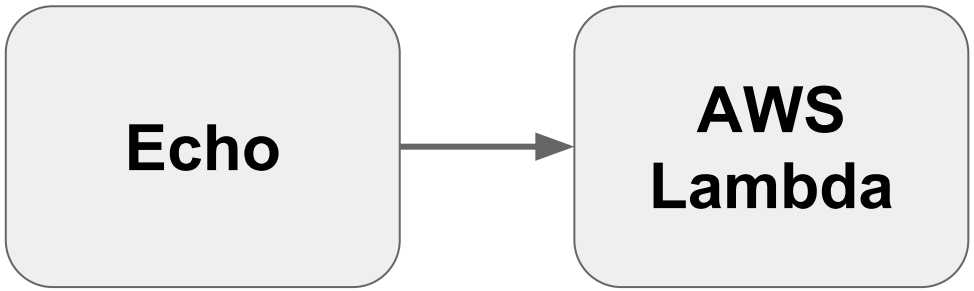
```
def ac_on(request):
    lg = subprocess.call(["irsend", "SEND_ONCE", "lg-ac", "BTN_1"])
    samsung = subprocess.call(["irsend", "SEND_ONCE", "samsung-ac", "BTN_1"])
    if lg != 0 or samsung != 0:
        return JsonResponse({'lg': lg, 'samsung': samsung}, status=500)
    return JsonResponse({})
```

에코



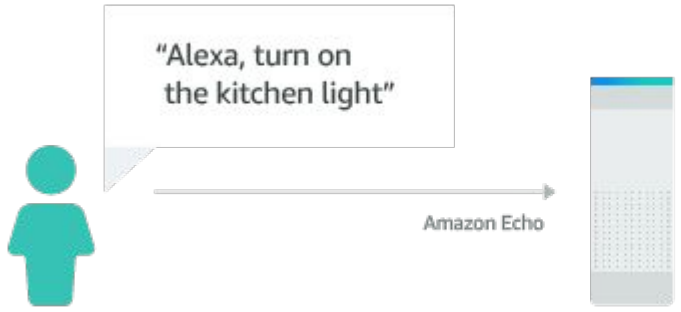
에코

- 에코가 요청한 명령 핸들러 구현
  - AWS Lambda라는 serverless computing service와 연동
- 다양한 연동방식
  - Custom Skills
  - Smart Home Skills
  - Flash Briefing Skills
  - Video Skills

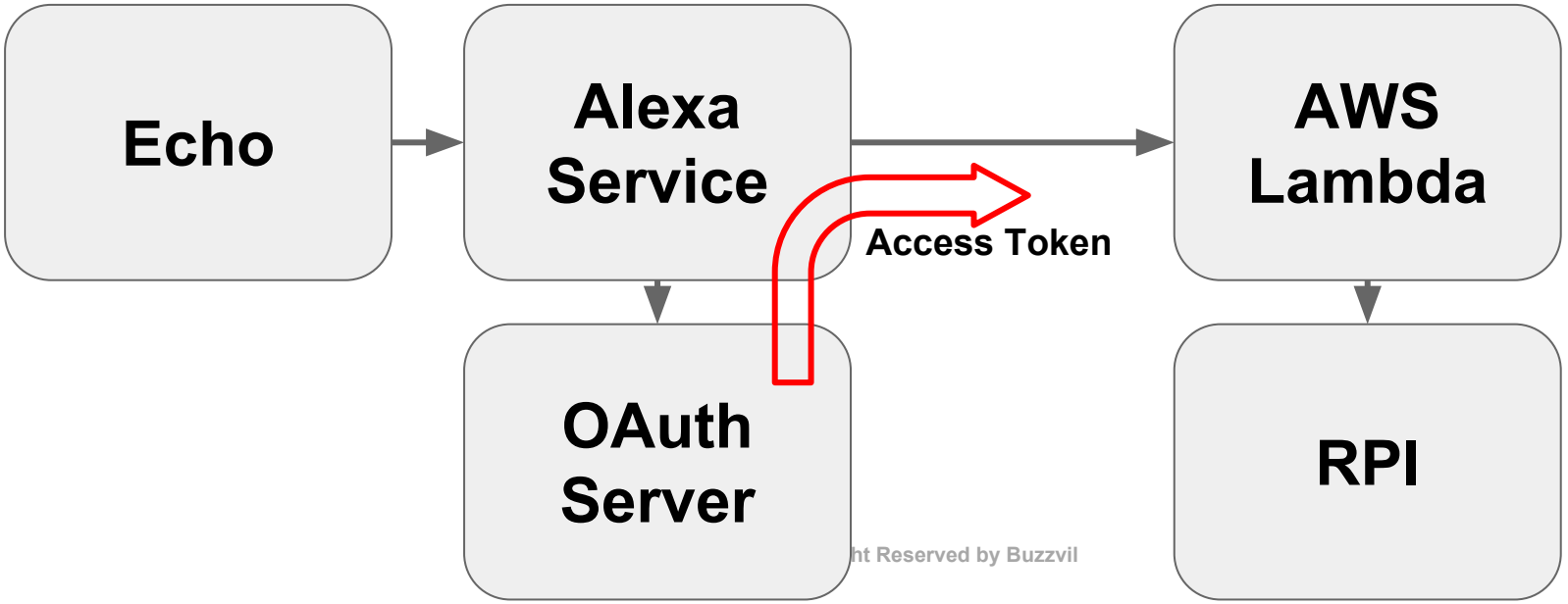


에코

- **Smart Home Skill API**
  - 클라우드 연동된 스마트 홈 디바이스 제어
  - 일반적으로 사용하는 명령들이 잘 정의되어 있음
    - TurnOnRequest
    - TurnOffRequest
    - SetTargetTemperatureRequest
  - Account Linking을 필수로 요구한다

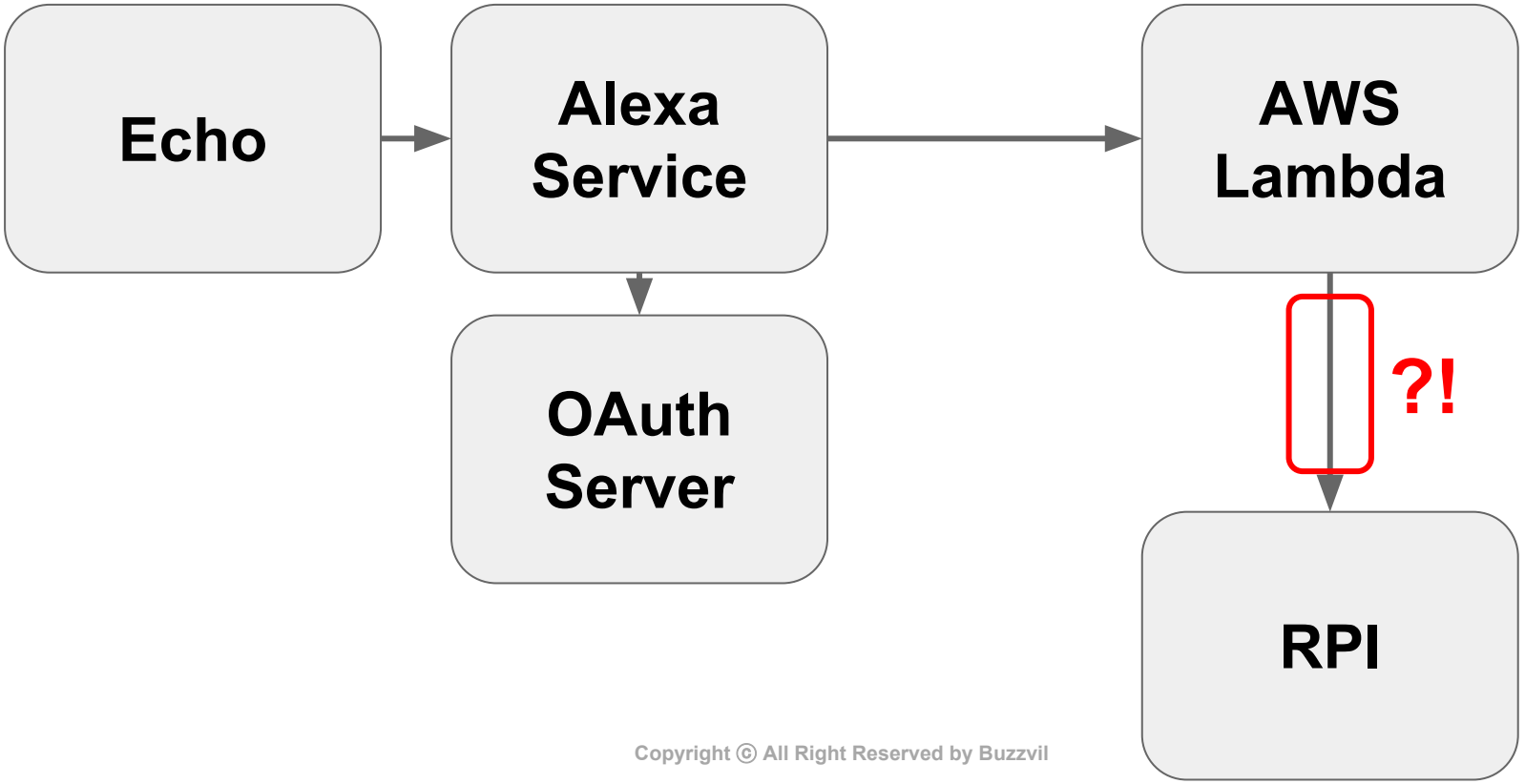


- **Account Linking**
  - OAuth 서버가 필요
  - 사용자가 에코를 구입하고 셋업하는 시점에 Alexa Service가 Access token을 받아옴
  - 팁: OAuth 서버를 직접 만드는 대신 페이스북 OAuth 서버 활용



Device discovery

- **Device discovery**
  - 디바이스는 공유기 내부 네트워크에 숨어있음



Device discovery

- **ngrok**
  - Secure tunnels to localhost
  - "I want to expose a local server behind a NAT or firewall to the internet."

```
jeseo$ ~/Downloads/ngrok http 8833
```

```
ngrok by @inconshreveable

Session Status                online
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://0745f90e.ngrok.io -> localhost:8833
Forwarding                    https://0745f90e.ngrok.io -> localhost:8833

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Device discovery

- **ngrok**
  - ngrok 실행시마다 도메인이 계속 바뀜
  - 유료 결제를 하면 고정 도메인 사용 가능

Device discovery

Basic	Pro	Business
$5 / mo per user, billed annually	$8.25 / mo per user, billed annually	$12 / mo per user, billed annually
(not available monthly)	($10 billed monthly)	($15 billed monthly)
	Everything in Basic, plus . . .	*Everything in Pro, plus . . .*
• Custom subdomains • Reserved domains • Google Apps SSO	• Whitelabel domains • Reserved TCP addresses • End-to-End TLS Tunnels	• IP whitelist tunnel access • Reserved wildcard domains • Priority Support
Per user limits	*Per user limits*	*Per user limits*
3 reserved domains	5 reserved domains	5 reserved domains
n/a	2 reserved TCP addresses	2 reserved TCP addresses

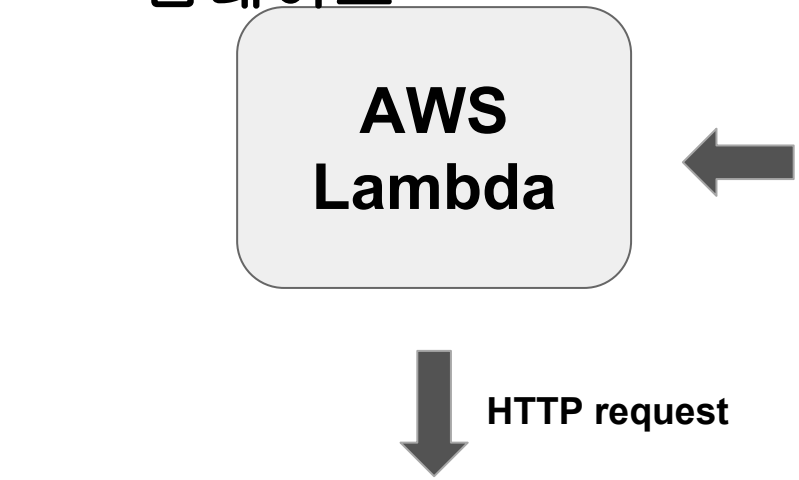
Device discovery

- **ngrok Client API**

```
jeseo$ curl -s http://localhost:4040/api/tunnels | python -m json.tool
{
    "tunnels": [
        {
            "config": {
                "addr": "localhost:8833",
                "inspect": true
            },
            "proto": "http",
            "public_url": "http://29518bb7.ngrok.io",
            "uri": "/api/tunnels/command_line+%28http%29"
        },
        {
            "config": {
```

Device discovery

- 주기적으로 public url을 가져와 DynamoDB table에 업데이트



machine_id	public_url
00000001234	http://d2.ngrok.io
00000002222	http://f3.ngrok.io

DynamoDB table

애플리Device discovery케이션

- **ngrok task**
  - requests/PynamoDB를 활용한 빠른 구현

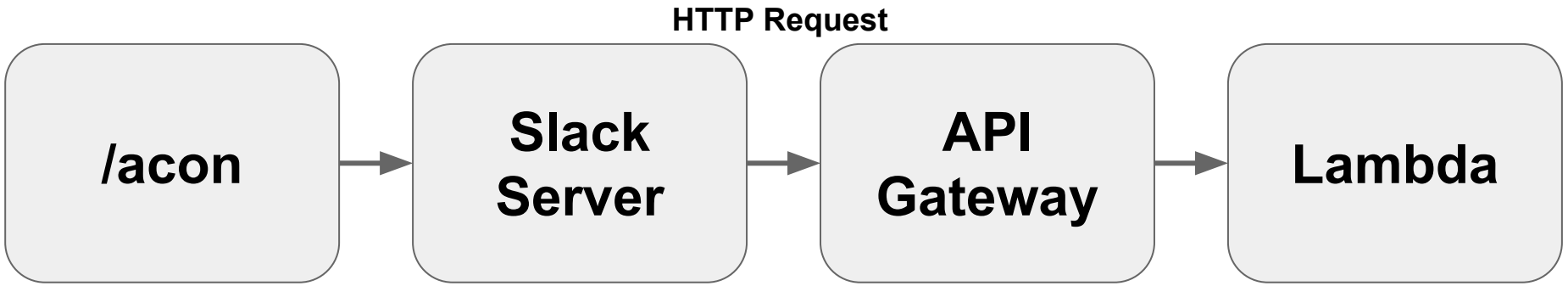
```
@shared_task
def update_public_url():
    response = requests.get('http://localhost:4040/api/tunnels')

    public_url = response.json()['tunnels'][0]['public_url']

    PublicUrl(
        machine_id=get_machine_id(),
        public_url=public_url,
    ).save()
```

Device discovery

- **비용 절약**
  - 유료 플랜 고정 도메인 3개 가격 한달에 5 불
  - DynamoDB 버지니아 기준 월 별 가격 = 0.47 불(1 write/sec) + 0.09 불(1 read/sec) = 0.56 불
  - ngrok Basic 플랜에 비해 1/10의 가격으로 무제한 도메인 사용 가능
  - 더 아끼고 싶으면 S3에 저장하자
  - 간단한 환경에서는 공유기의 DDNS/포트 포워딩 기능 이용

Slack 연동

- Slack 연동
  - Slash Commands
    - /acon - 에어컨 끄기
    - /acoff - 에어컨 켜기
    - /acxxx - 온도 조절
  - HTTP 요청을 통해 Lambda를 실행하기 위해 API Gateway 사용

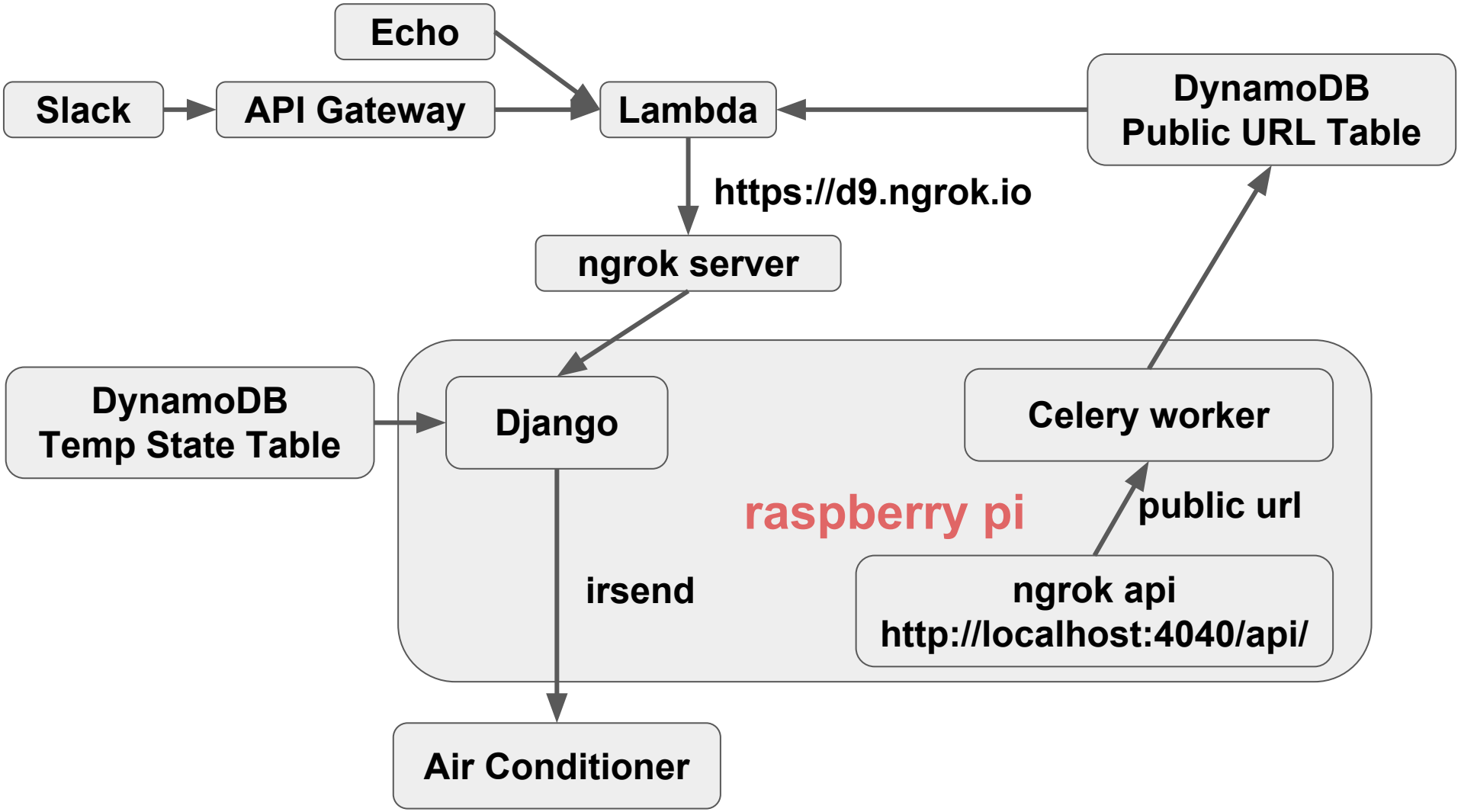


Slack 연동

- **Temperature state aware command**
  - 에어컨 ON명령에는 온도값도 항상 포함이 되어있어야 한다
  - /acon를 /acwarm, /acmedium, /accool 로 변환이 필요
  - /acwarm, /acmedium, /accool 중 마지막에 호출된 커맨드를 저장해야함
  - DynamoDB에 상태값을 저장

```
def ac_on(request):
    state_button_map = {
        'warm': 'BTN_3',
        'medium': 'BTN_5',
        'cool': 'BTN_8',
    }

    config = six.next(ACConfig.query(hash_key=settings.AC_LOCATION))
    btn_name = state_button_map[config.state]
    return ac_command(btn_name)
```

최종 설계도



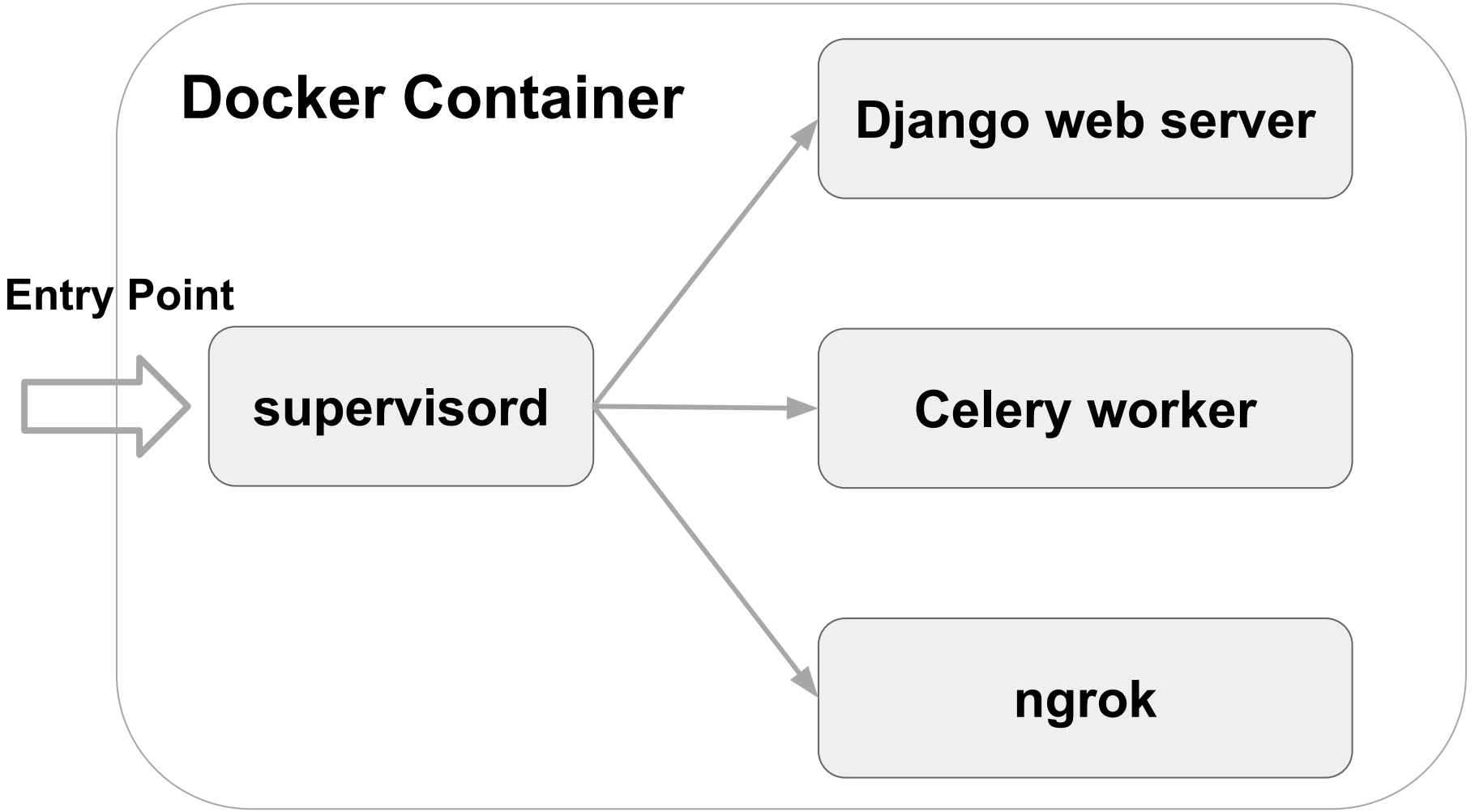
배포 자동화

- **AC controller application을 설치하기 위한 단계**
  - IR 핀 설정
  - ngrok 설치
  - pip 패키지 설치 및 django project 소스코드 복사
  - supervisord를 이용해 ngrok, django web server, celery worker daemonize

배포 자동화

- 도커
  - 쉽게 말하면 윈도우의 포터블 앱과 비슷
  - 컨테이너 안에 필요한 애플리케이션을 설치하고 이미지를 만들어 라즈베리파이에서는 단순히 이미지를 받아 실행

배포 자동화



배포 자동화

- **Ansible**
  - 서버 셋업 자동화 툴
  - **Puppet, Chef** 등과 비슷하나 **Agent**가 필요없고 **SSH** 프로토콜 사용

```
---
- name: Setup raspberry pi
  hosts: all
  gather_facts: true

  roles:
  - {role: ir, gpio_in_pin: 15, gpio_out_pin: 14}
  - {role: docker}
  - {role: ac,
      AC_LOCATION: 3rd,
      AWS_ACCESS_KEY_ID: x,
      AWS_SECRET_ACCESS_KEY: x,
      SLACK_TOKEN: x,
      SLACK_URL: "http://yourdomain.com"
    }
```

배포 자동화

- **업그레이드된 AC controller setup하기**
  - ansible 명령으로 application 설치
    - **ansible-playbook -i hosts setup_rpi.yml**
  - 완성!

One more thing

One more thing

- **조도 센서를 이용한 자동 어에컨 끄기**

❑ 조도 센서를 폴링하여 조명이 on -> off 상태로 변경이 됐는지 체크

❑ 마지막 조명 on/off 상태값을 저장해야함

❑ 상태값은 sqlite3 db에 저장하고 Django ORM을 통해 접근

```
@shared_task
def check_light():
    is_light_on = LightSensor().is_light_on()
    last_light_status = LightStatus.objects.get(pk=1)

    if last_light_status.is_light_on != is_light_on:
        if not is_light_on:
            requests.post(os.environ['SLACK_URL'], {
                'token': os.environ['SLACK_TOKEN'],
                'command': '/acoff',
            }, timeout=30)

        last_light_status.is_light_on = is_light_on
        last_light_status.save()
```

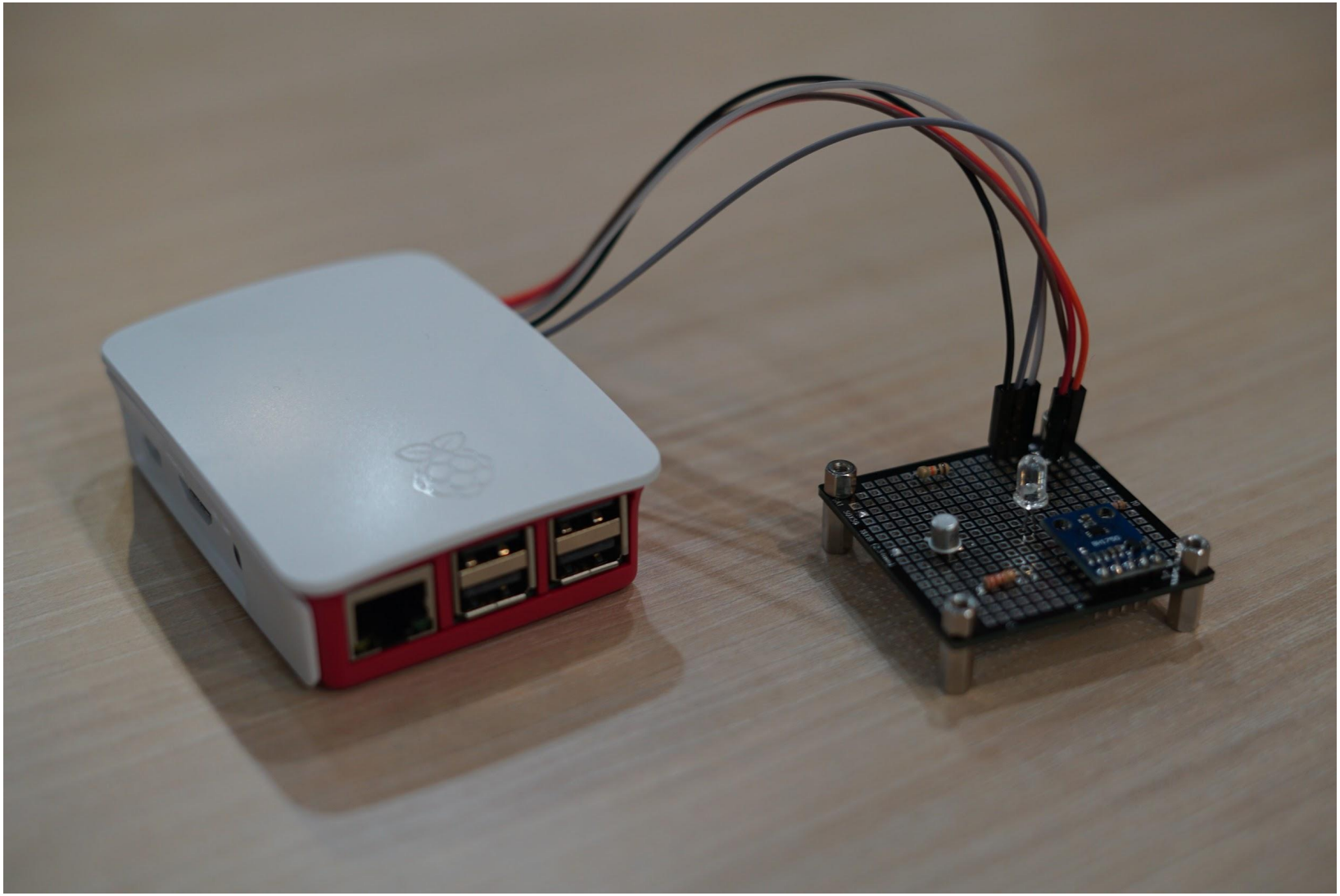
Summary

- **재료비**
  - 라즈베리파이3 + 공식케이스 + 방열판 = 53,350원
  - SD카드 = 4,410원
  - 트랜지스터 = 700원
  - 적외선 LED = 400원
  - 저항 = 100원
  - 만능기판 = 1,300원
  - 만능기판 다리 = 100원
  - 점퍼 케이블 = 300원
  - 총 = 60,660원
- **라즈베리파이 zero W 가 나왔습니다**
  - 해외 판매가 10불

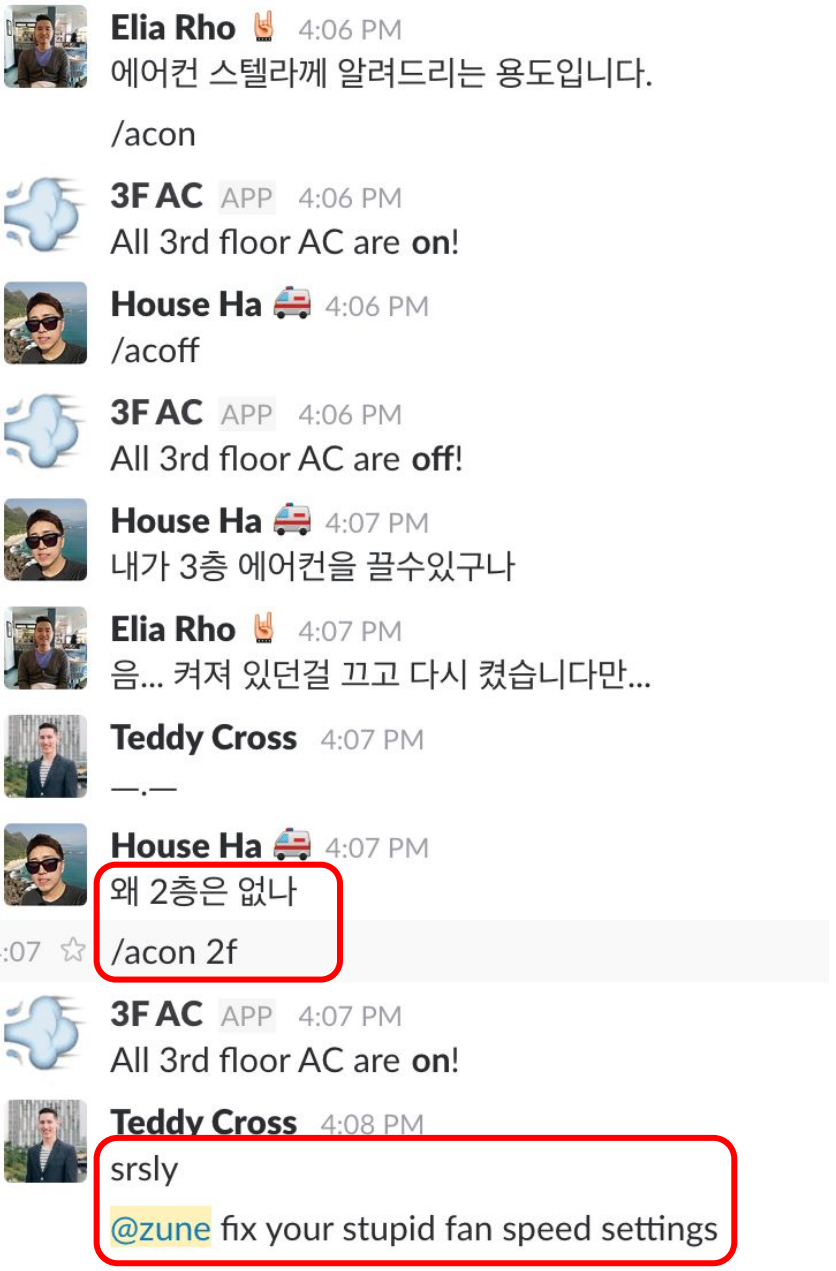
Summary

- **납땜을 위한 준비물 비용**
  - 라즈베리파이 입출력 키트 = 14,600
  - 테프론 와이어 AWG30 = 5,380원
  - Kester 유연납 1.0mm /50g = 4,400원
  - 멀티미터 FLUKE-101 = 43,500원
  - HAKKO FX-888D 디지털인두기 = 143,550원?!
  - 교체용인두팁 T18-K = 11,880원
  - 니퍼 = 집에 있는거
  - 총 = 223,310원
- **에코**
  - 에코닷 = 55,000원

Summary



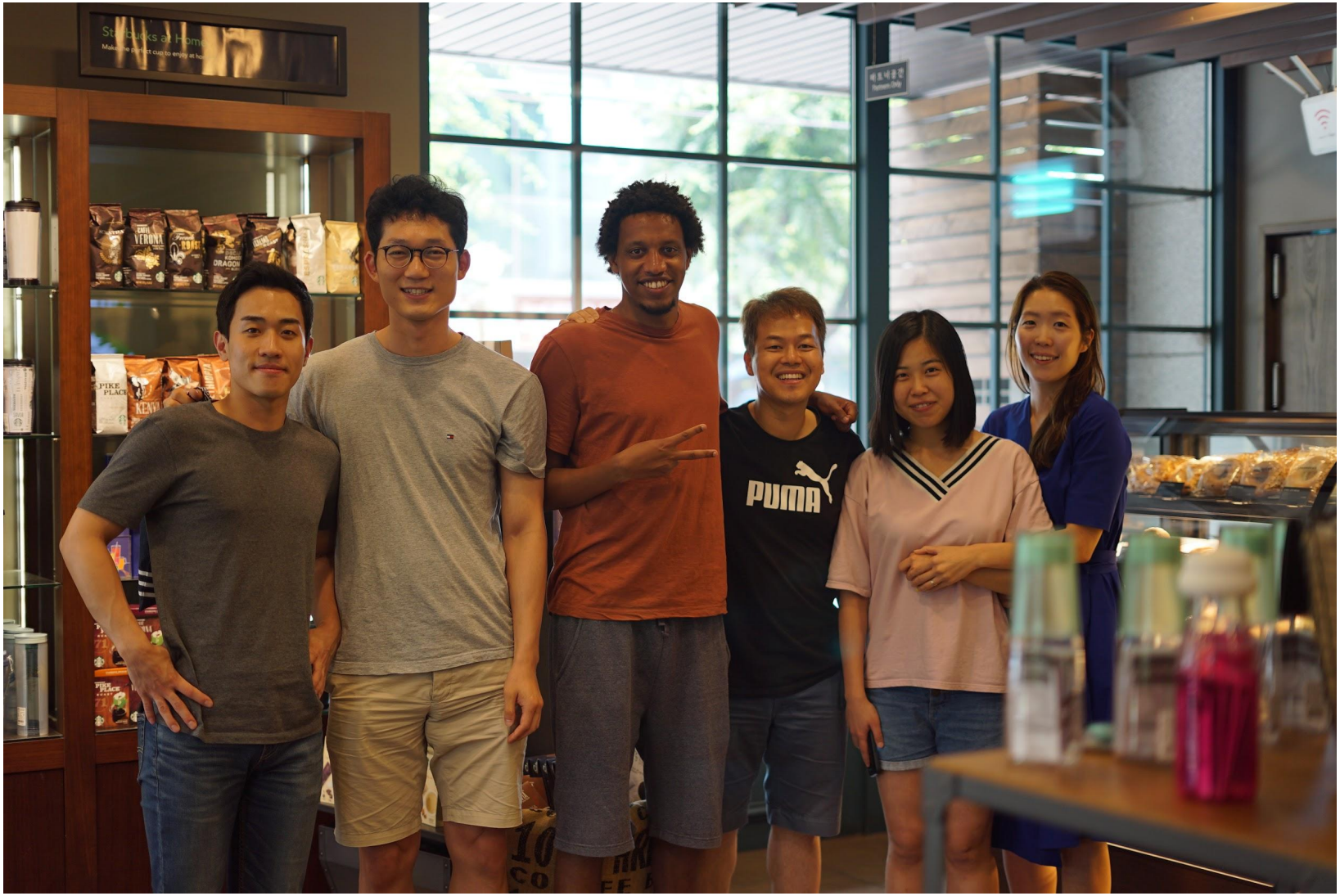
Summary


Elia Rho 4:06 PM
에어컨 스텔라께 알려드리는 용도입니다.
/acon
3F AC APP 4:06 PM
All 3rd floor AC are on!
House Ha 4:06 PM
/acoff
3F AC APP 4:06 PM
All 3rd floor AC are off!
House Ha 4:07 PM
내가 3층 에어컨을 끌수있구나
Elia Rho 4:07 PM
음... 켜져 있던걸 끄고 다시 켰습니다만...
Teddy Cross 4:07 PM
—.—
House Ha 4:07 PM
왜 2층은 없나
4:07
/acon 2f
3F AC APP 4:07 PM
All 3rd floor AC are on!
Teddy Cross 4:08 PM
srsly
@zune fix your stupid fan speed settings

Summary

- 마무리하며
  - **Python/Django/Celery**를 활용해 필요한 애플리케이션 로직을 빠르게 개발
    - **HTTP** 서버 구축
    - **ngrok** 무료 사용
    - 에어컨 온도 설정상태 보관
    - 조도센서 상태 변경 캐치
  - **Django/Celery** 조합을 활용해 하나의 프로젝트로 서버부터 주기적인 태스크까지 다양한 일을 수행
  - **Python**을 활용해 빠르게 프로토타이핑을 하고 시간이 많이 걸리는 다른 부분에 집중할 수 있었음

Thanks to



감사합니다

Q&A

The project is open-sourced

https://github.com/Buzzvil/hardware-lab