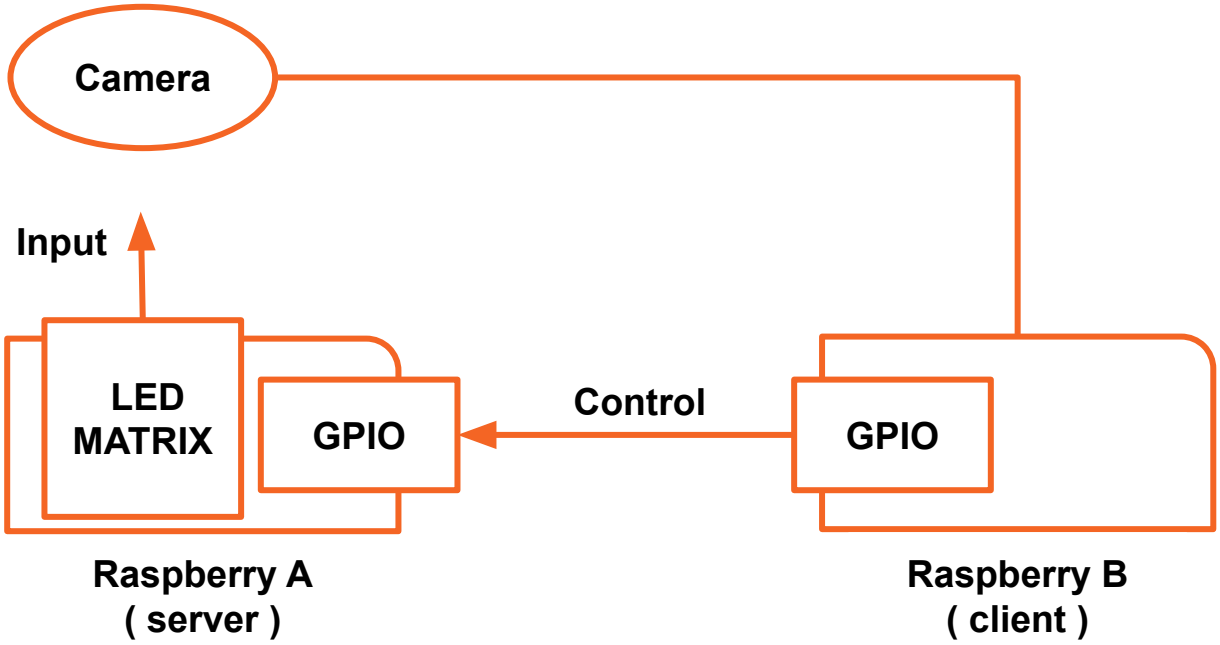
임베디드 시스템
최종발표

17011505 안창언

Index

1. 소개

플로우차트


Camera
Input
LED
MATRIX
GPIO
Control
GPIO
Raspberry A
(server)
Raspberry B
(client)

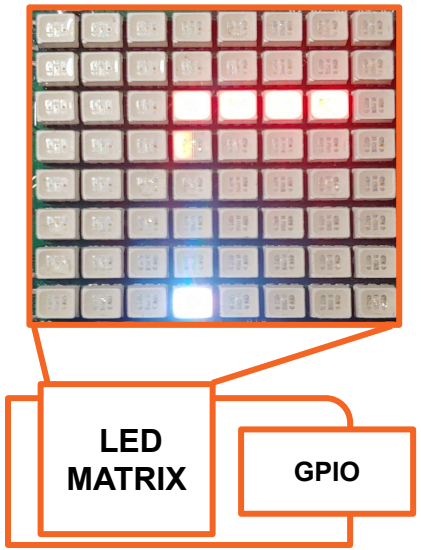
2. 특성

임베디드 특성

1. 카메라 인풋을 리얼타임으로 분석
2. 적절한 액션을 GPIO 핀으로 아웃풋
3. LED 매트릭스로 표출

3. 구현

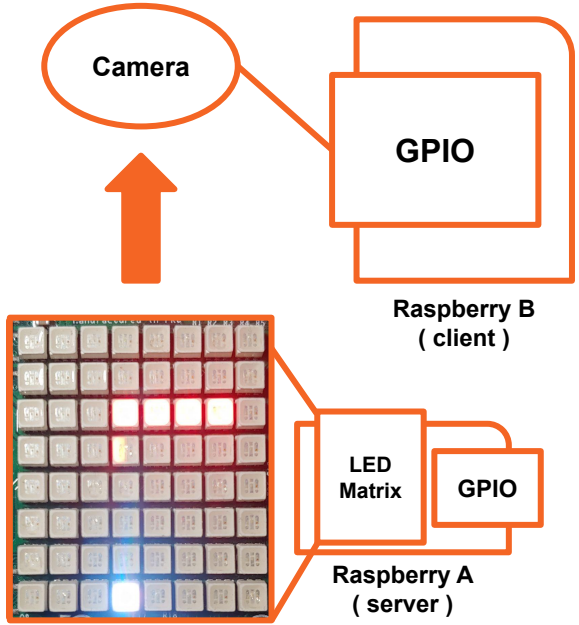
1. 텍스트 게임 표출




```
class Display:
    def __init__(self, **kwargs):
        self.arr = [["0" for x in range(0, 8)]

        if kwargs["type"] == "curses":
            self.stdscr = kwargs["stdscr"]
        elif kwargs["type"] == "sense":
            self.sense = SenseHat()
        elif kwargs["type"] == "both":
            self.stdscr = kwargs["stdscr"]
            self.sense = SenseHat()

    def set_arr(self, ship, obstacles):...

    def print_curses(self):...

    def print_sense(self):...

    def print_input(self, action):...

    def wait_input(self):...
```

3. 구현

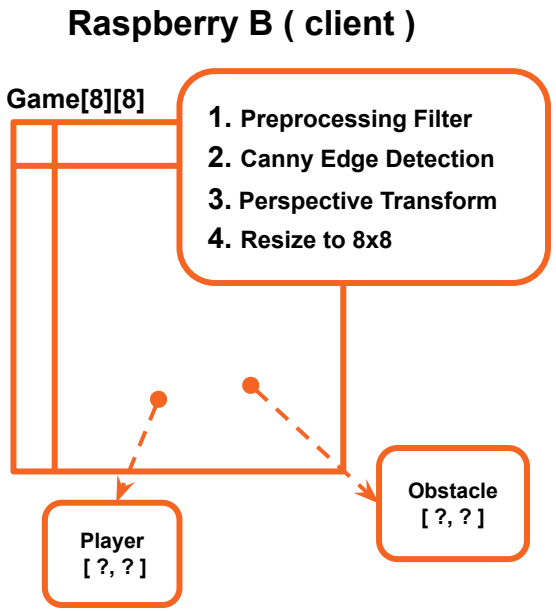
2. 카메라 인풋




```python
class Camera:
    def __init__(self):
        self.cnt = 0

        self.camera = PiCamera()
        self.camera.resolution = (640, 480)
        self.camera.framerate = 32
        time.sleep(0.1)

        self.rawcap = PiRGBArray(self.camera,

    def get_frame(self):...
```

3. 구현

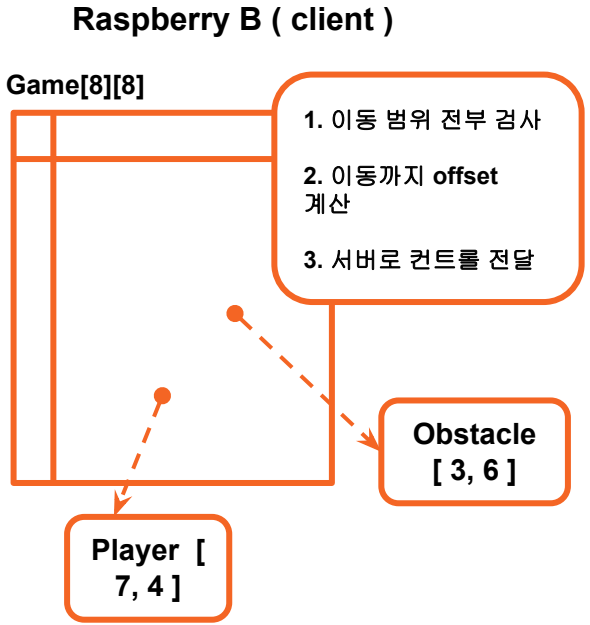
3. 이미지 데이터 프로세싱




```python
class ImageScanner:
    def __init__(self):
        self.the_orig = None
        self.orig = None
        self.edged = None
        self.ratio = None
        self.warp = None
        self.pixel = None

    def scan_frame(self, image, cnt):...

    def preprocess(self, image):...

    def dectect_contour(self, image):...

    def find_contour(self, cnts):...

    def warp_contour(self, screenCnt):...

    def crop(self, warp, h_start=19, h_end=20

    def pixelize(self, warp):...
```

3. 구현

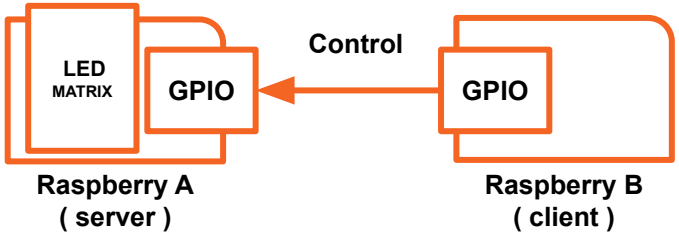
4. 게임 액션 판별

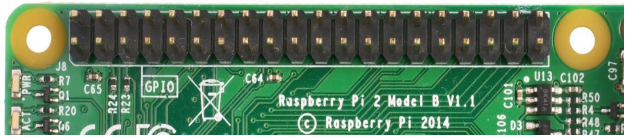



```python
class Actor:
    def __init__(self):
        self.prev_matrix = [[0 for x in range(0, 8)
        self.prev_ship_pos = 0

        self.curr_matrix = [[0 for x in range(0, 8)
        self.curr_ship_pos = 4

        self.action_queue = []
        self.queue_len = 0

    def locate_elements(self, scanned):...

    def get_path(self, x):...

    def get_action(self):...

    def queue_action(self, offset):...

    def verify_action(self):...

    def pass_turn(self):...
        #self.action_queue.clear()
```

3. 구현

5. 컨트롤 신호 전달

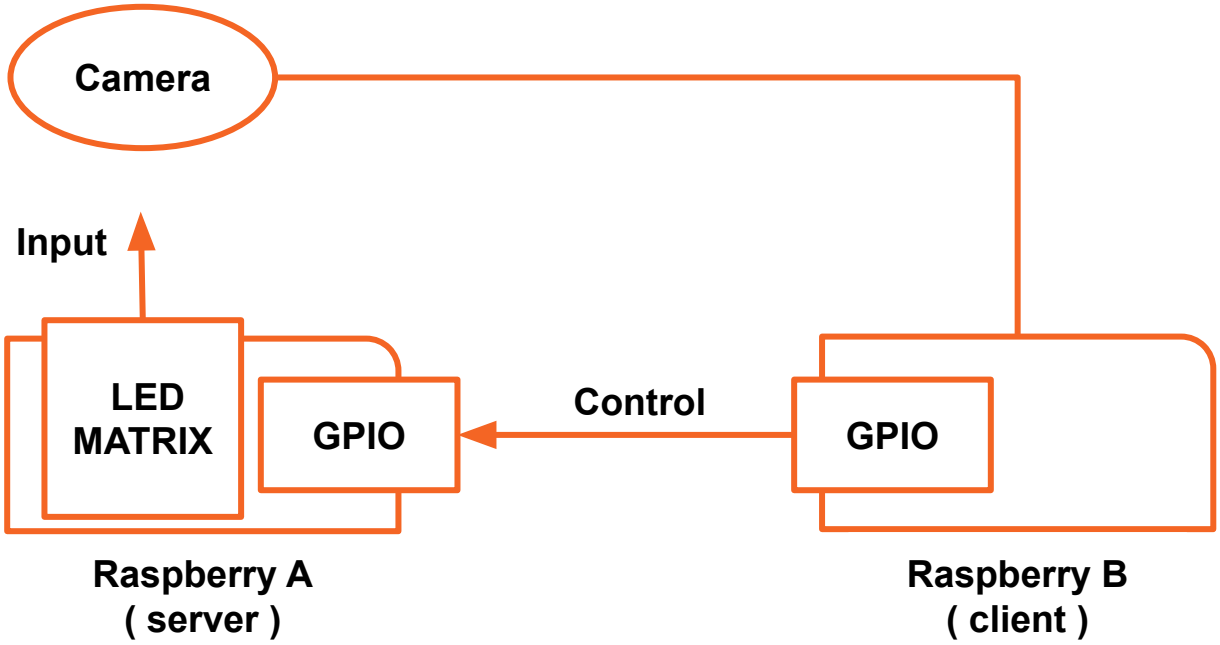





```python
class Input:
    def __init__(self, **kwargs):
        self.type = kwargs["type"]

        if self.type == "curses":
            self.source = kwargs["stdscr"]
        elif self.type == "serial":
            self.ser = serial.Serial("/dev/
        elif self.type == "both":
            self.source = kwargs["stdscr"]
            self.ser = serial.Serial("/dev/

    def get_input(self):...

    def get_input_curses(self):...

    def get_input_serial(self):...
```

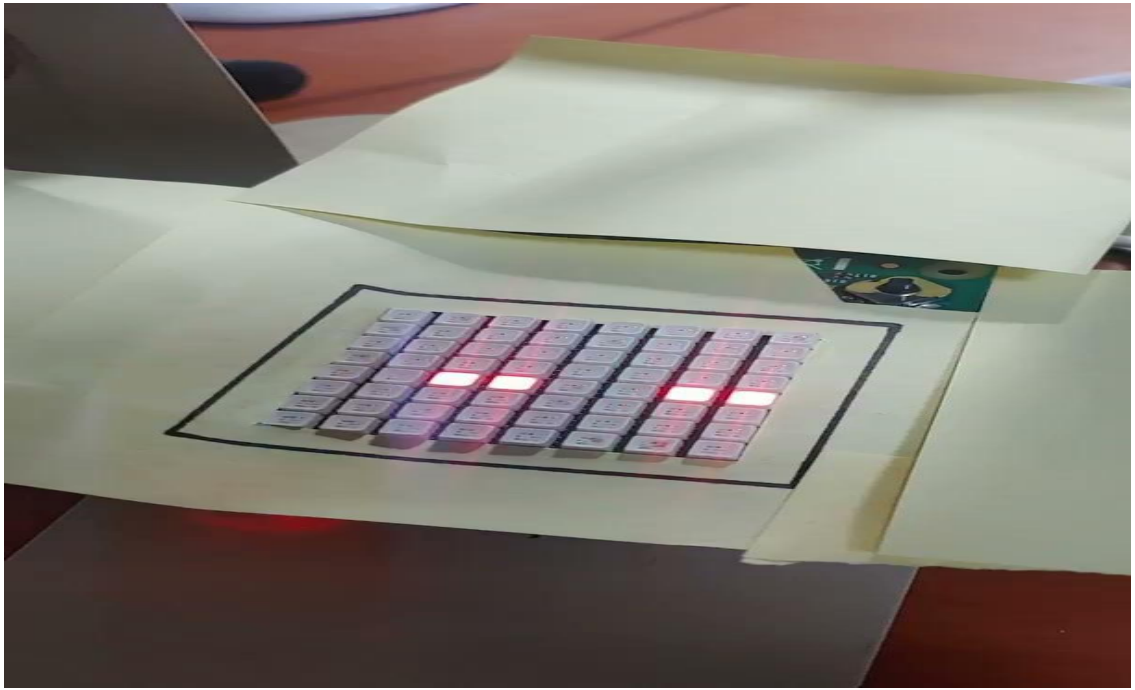
4. 데모

1. 데모 시나리오


Camera
Input
LED
MATRIX
GPIO
Control
GPIO
Raspberry A
(server)
Raspberry B
(client)

4. 데모

2. 데모 비디오



5. 목표

달성 목표

매우 우수

카메라 데이터 인식과 두 라즈베리 파이 간 통신 프로토콜을 직접 구현

우수

카메라 데이터 인식과 두 라즈베리 파이 간 통신을 **UART**로 구현

보통

카메라 데이터 인식은 구현했지만 두 라즈베리 파이 간 통신 불가능, 한개의 보드에서만 구동

실패

카메라 데이터 인식과 두 라즈베리 파이간 통신 불가능, **LED** 매트릭스에 텍스트 게임만 표출

감사합니다.