Embedded System Software HW #4

due date: 2019. 6. 21.

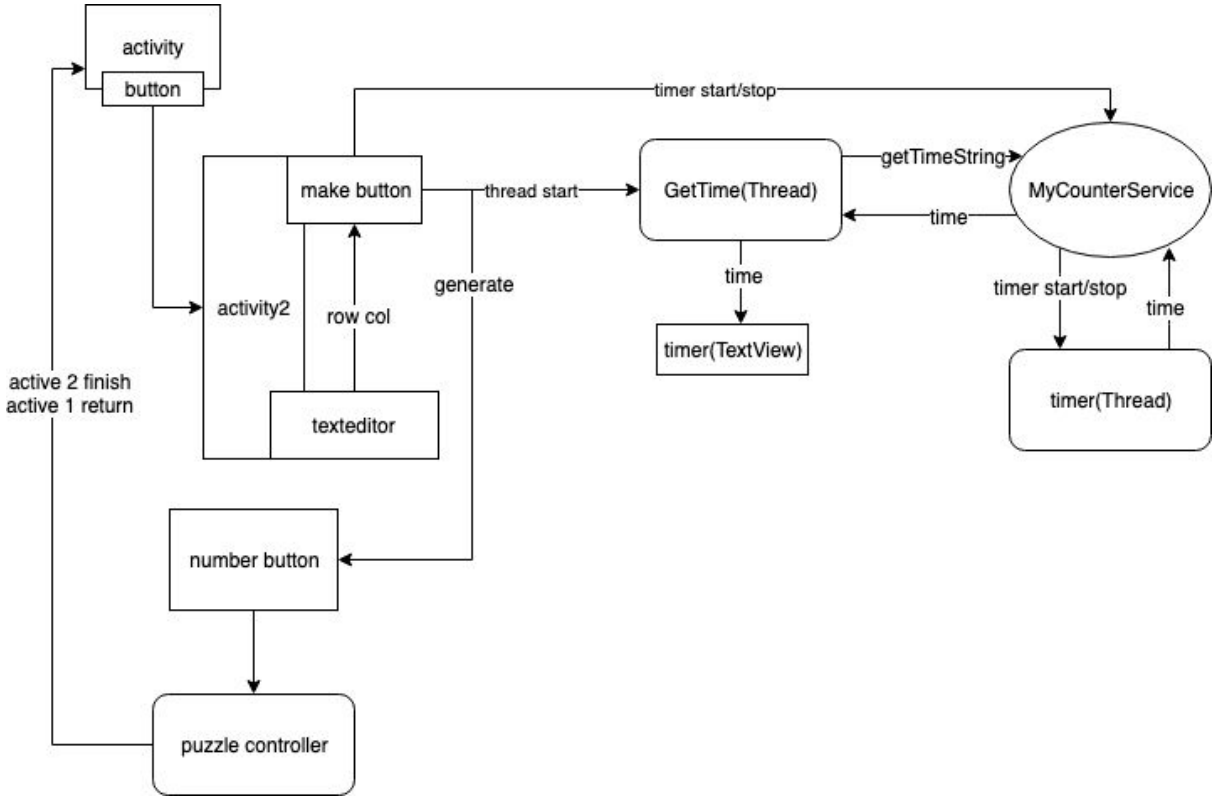
student name: 윤제형
student id: 20151575

목차

1. 목표

- 임베디드 환경에서 appliction 계층을 담당하고 있는 android platform을 이해하고 activity 를 활용하여 간단한 app을 제작하여 본다. 이 과정에서 android service를 제작하여보고 적용하여 본다.
- android service란 service manager에 등록되는 service로 app의 back ground 에서 실행되어지는 service를 말한다.

2. 순서도



- 첫번째 activity는 학번과 퍼즐 게임을 실행하는 버튼으로 이루어져 있다. 이 버튼을 누르면 activity2라고 정의된 puzzle game을 실행한다 puzzle game은 버튼과 time버튼 text 입력창으로 이루어져 있는데 text 입력을 통해 row, colum을 입력받아 number buttons 들을 생성한다. 이와 동시에 mycounterservice를 bind하여 timer 를 실행시킨다. 이후 service 호출하여 지속적으로 timer를 갱신하는 thread 를 생성하여 time을 갱신한다. puzzle 이 solve되었다면 초기 activity로 돌아간다.

3. 구현

```
26          OnClickListener listener=new OnClickListener(){
27              public void onClick(View v){
28                  Intent intent=new Intent(MainActivity.this, MainActivity2.class);
29                  startActivity(intent);
30              }
```

- 2번째 activity를 생성하는 버튼의 listener이다.

```
224         OnClickListener make_listener=new OnClickListener(){
225             public void onClick(View v){
226                 String temp=data.getText().toString();
227                 String datas[] = temp.split(" ");
228                 row = Integer.parseInt(datas[0]);
229                 col = Integer.parseInt(datas[1]);
230                 initButtons();
231                 makeButtons();
232                 stopTimer();
233                 startTimer();
234             }
```

- button을 생성하는 make button에 붙은 listener로 입력받은 row col에 해당하는 버튼들을 생성하고 service의 timer와 이를 계속 가져와 textview에 update하는 thread를 실행시킨다.

```
private void makeButtons(){
    btns = new Button[row*col];

    for(int i= 0 ; i < row; i++){
        LinearLayout test = new LinearLayout(this);
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(LinearLayout.
LayoutParams.MATCH_PARENT,LinearLayout.LayoutParams.MATCH_PARENT,1);
        test.setLayoutParams(params);
        test.setOrientation(LinearLayout.HORIZONTAL);

        for(int j= 0 ; j < col; j++){
            btns[i*col+j] = new Button(this);
            btns[i*col+j].setWidth((int)widthDp/col);
            btns[i*col+j].setHeight((int)heightDp/row);
            btns[i*col+j].setText(Integer.toString(i*col+j+1));
            btns[i*col+j].setOnClickListener(button_listener);
            test.addView(btns[i*col+j]);
        }

        buttonLinear.addView(test);
    }
```

- make button에 의해 생성된 puzzle button들은 linearlayout을 통해 격자모양으로 정렬되고 layout에 추가된다.

```
ServiceConnection conn = new ServiceConnection(){

    public void onServiceConnected(ComponentName name, IBinder service){
        MyBinder mb = (MyBinder) service;
        ms = mb.getService(); // 서비스가 제공하는 메소드 호출하여
        isService = true;
    }
    public void onServiceDisconnected(ComponentName name) {
        isService = false;
    }
};
```

- bind service가 이루어질시 시행되는 connect 함수이다. 이를 통해 mycounterservice의 binder에 접근 가능해진다.

```
    private class GetTime implements Runnable {
        private Handler handler = new Handler();
        public void run() {
            while (running) {
                if(ms == null) {
                    continue;
                }
                 handler.post(new Runnable() {
                    @Override public void run() {
                        try {
                            timetext.setText(ms.getTimeString());
```

- puzzle game에서 실행되는 thread로 주기적으로 service를 통해 timer count를 넘겨받아 화면의 textview에 이를 출력한다.

```
public class MyCounterService extends Service{

    int count;
    Timer timer = null;
    IBinder mBinder = new MyBinder();
    Thread thd;
    class MyBinder extends Binder {
        MyCounterService getService() { // 서비스
            return MyCounterService.this;
        }
    }
```

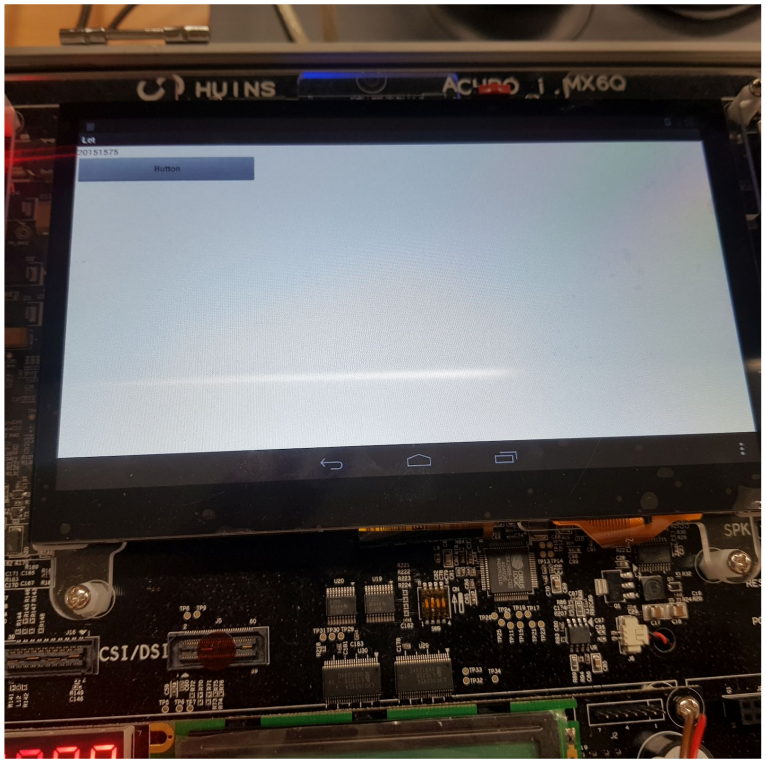
```
@Override
public IBinder onBind(Intent intent) {
    // 액티비티에서 bindService() 를 실행하면
    // 리턴한 IBinder 객체는 서비스와 클라이언
    return mBinder; // 서비스 객체를 리턴
}
public boolean onUnbind(Intent intent) {
    return super.onUnbind(intent);
}
```

- 추가구현사항인 mycounterservice로 2번째 사진과 같이 binder를 통해 activity에 bind 되어 timer data를 리턴한다.

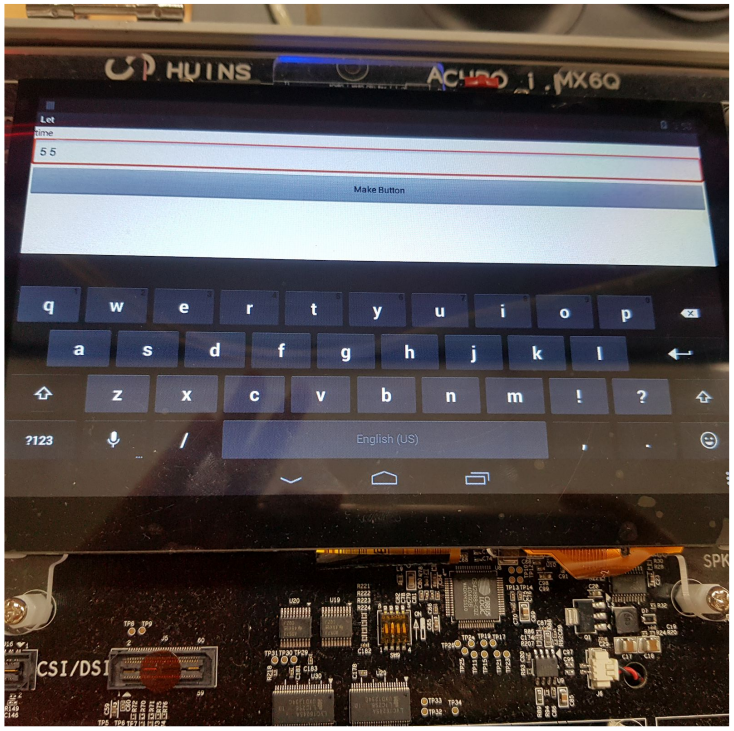
```
private class Timer implements Runnable {

    boolean running = true;
    public void run() {
        running = true;
        while (running) {
            try { count++;Thread.sleep(100); } catch (InterruptedException e) { e.
printStackTrace(); }
        }
```

- timer service에서 실행되는 thread로 0.1 초마다 count를 갱신한다.

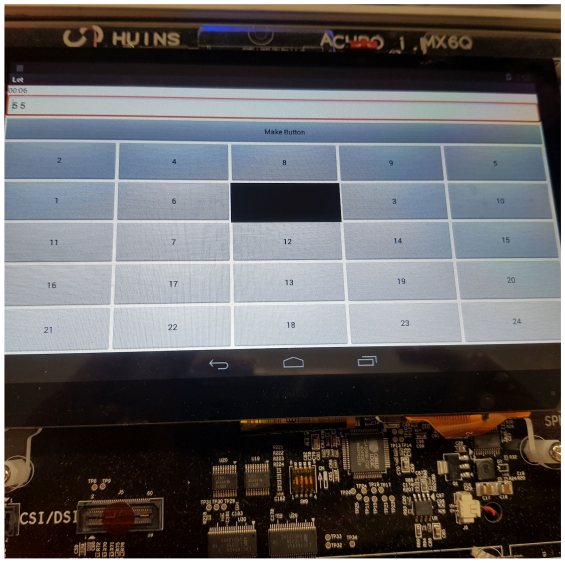
4. 실험



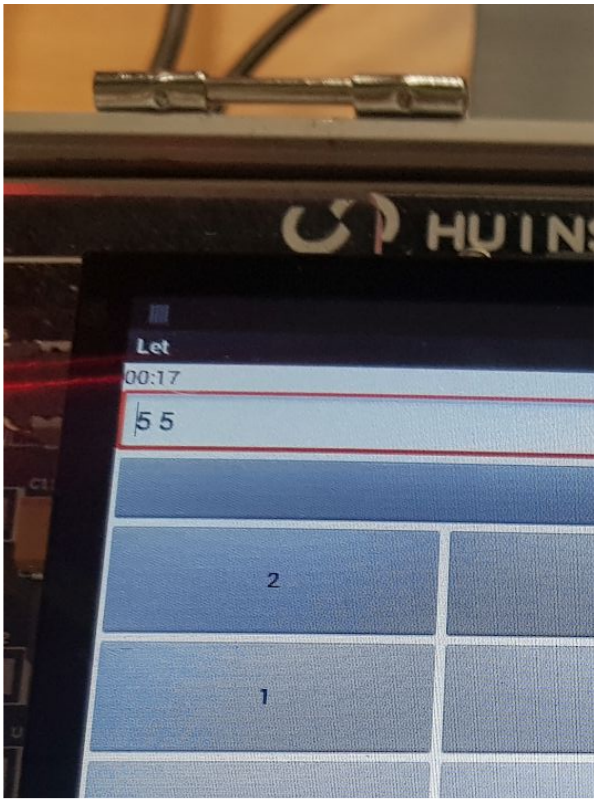
- 초기 상태의 activity1



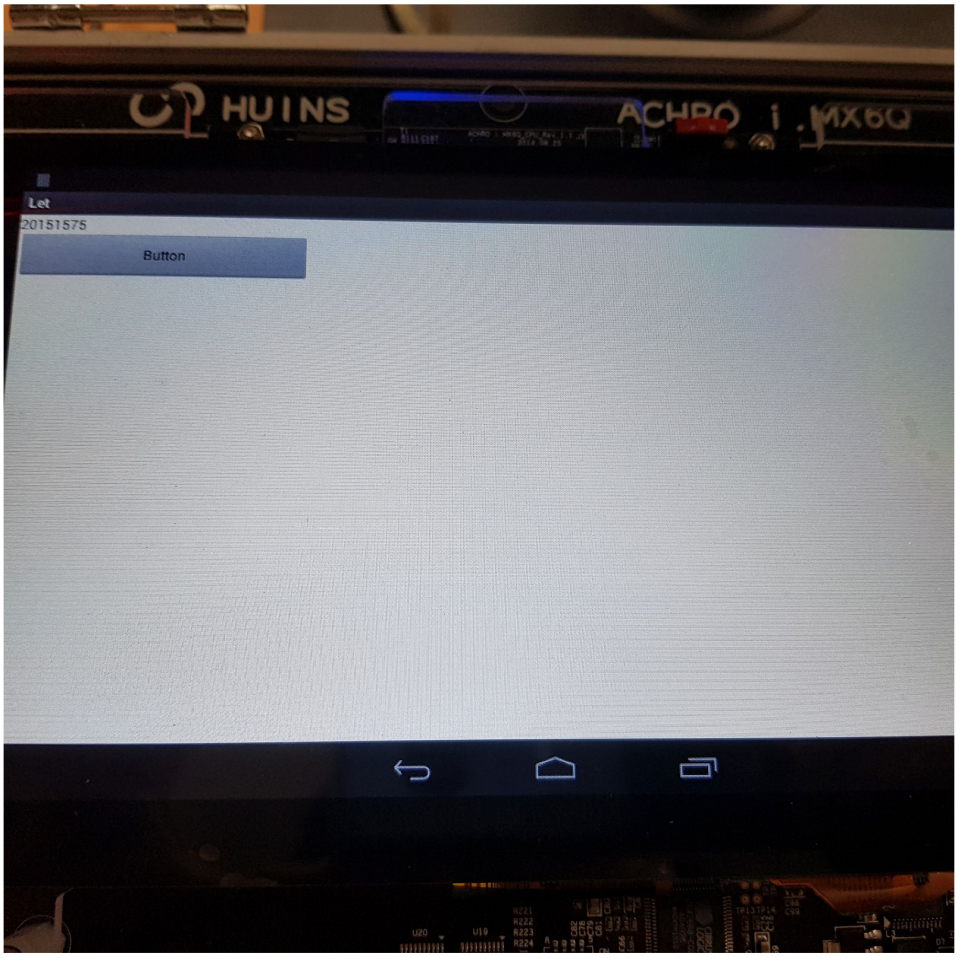
- buttons을 눌러 생성된 activity2 이를 make button을 통해 puzzle을 생성할수 있다



puzzle이 생성된 상태.



- 게임이 진행되는 동안 추가구현인 timer가 작동하는 모습



-puzzle이 solve 되고 초기상태인 activity1으로 회귀한 상태