

# Rubik Solver

## HIGH SCHOOL FINAL EXAM PROJECT

*A Rubik Solver Robot using two robotic arms, Arduino and an Android Smartphone.*

**Stagi Roberto VAI**

---

2015

---

## Rubik Solver

---

### Generalities

The project is based on two mechanical arms, each composed of one servo and a grip. The two arms forms a 90° angle and have to “take” the cube, performing all the moves.

The Android Smartphone will be used to take the pictures of each side of the cube and to perform the solving algorithm (the efficient Kociemba's 2Phase algorithm). *For furhter information about the solving algorithm, please visit [www.kociemba.org](http://www.kociemba.org).*

The Smartphone and the Arduino board will communicate through a Bluetooth module.

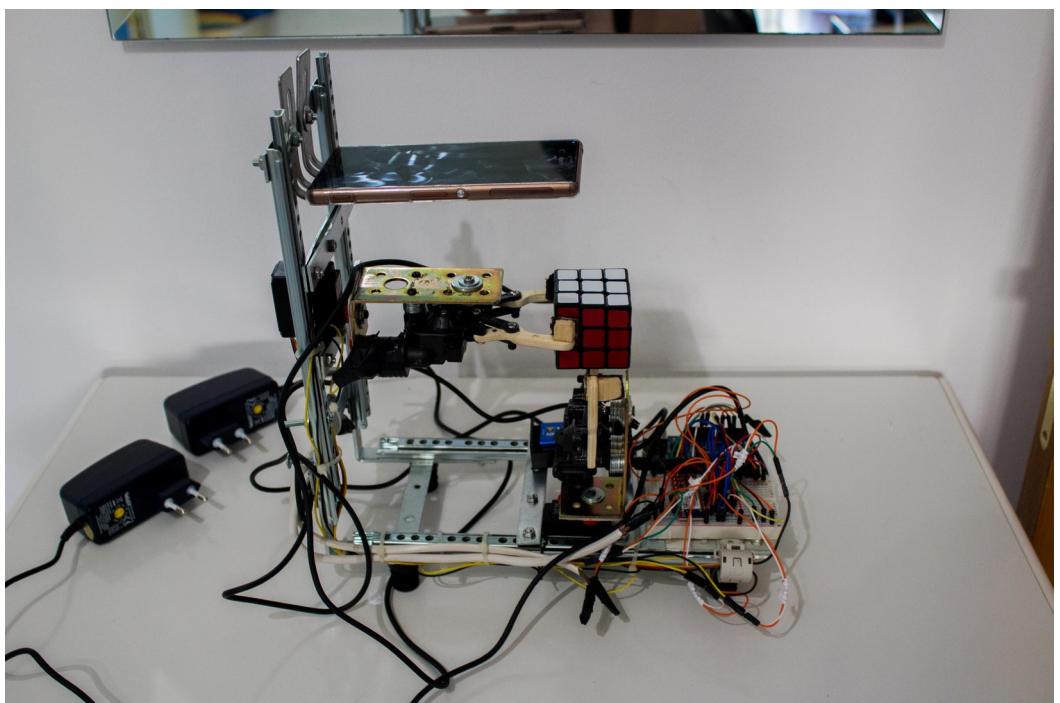


Figura 1 Struttura dei bracci collegati ad Arduino. Si può vedere anche la posizione del telefono per poter fare le foto alle facce del cubo.

## Notation

The six sides are called: Up, Down, Back, Front, Right, Left.

To indicate the moves for each side, we'll use the following method:

- A clockwise 90° rotation is indicated with the initial letter of the side's name.  
For example: **U** (up)
- A counter-clockwise 90° rotation is indicated with the initial letter of the side's name followed by an apostrophe. For example: **U'**
- A 180° rotation is indicated with the letter followed by a 2: **U2**

All moves are considered watching to the side from the upper face.

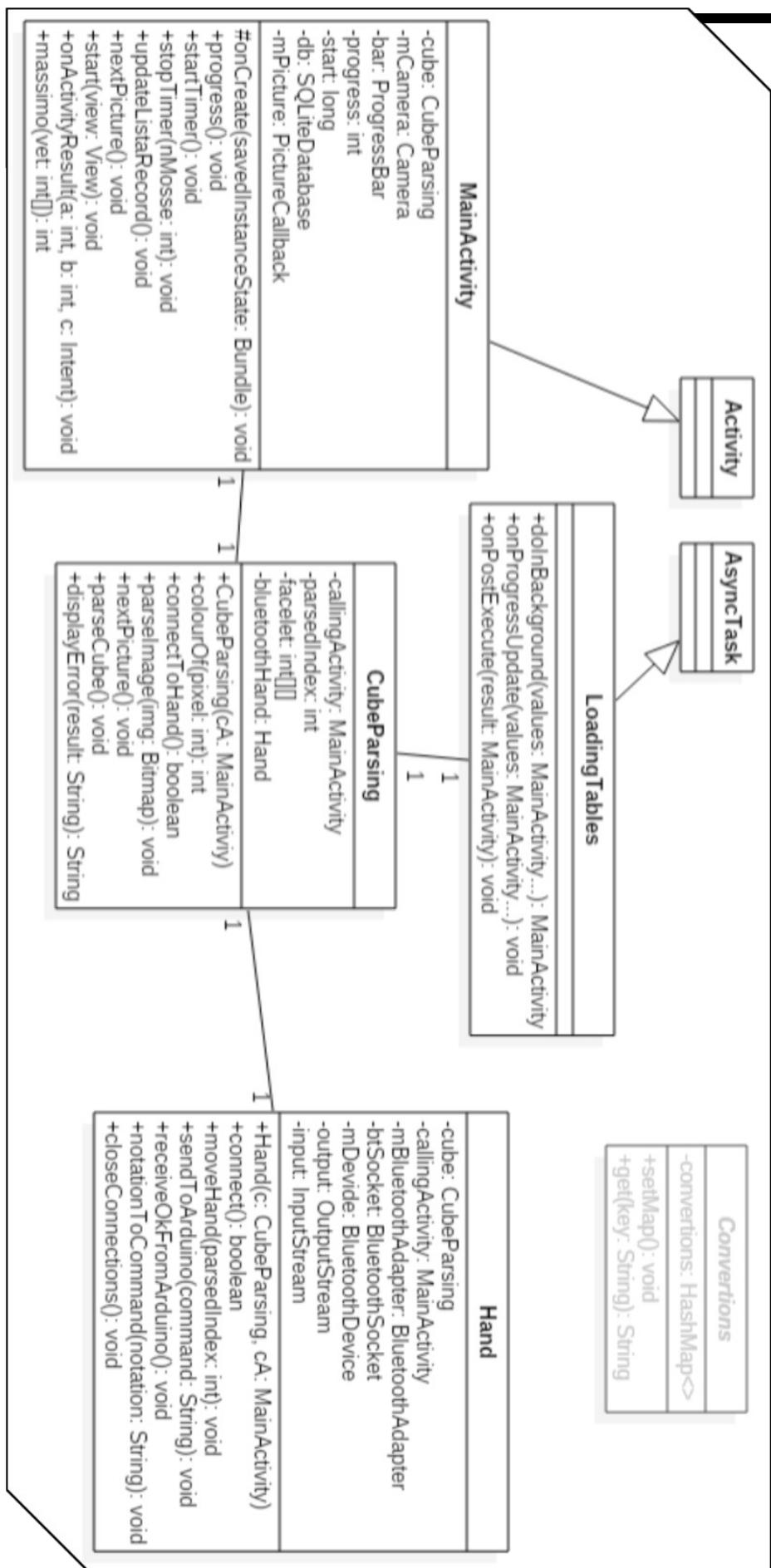
For example, if we consider the "back" side, we have to rotate the cube and see the back side as the upper one.

## Diagrams

The Android application have the following classes

- **MainActivity**: App interface. It has a button to start the solving procedure and a progress bar.
- **CubeParsing**: It analyzes the cube and get the solving procedure from the 2Phase algorithm.
- **LoadingTables**: It loads the tables necessary for the 2Phase algorithm.
- **Hand**: It is the interface with the Bluetooth module and Arduino.
- **Conversions**: Abstract class for notations.

# Rubik Solver



# Rubik Solver

The App has a very simple SQLite database to store the records (in terms of time).



Figura 2 Diagramma concettuale del database.

The Arduino sketch has two classes to interface with the Servo (wrist) and the DC Motor (grip).

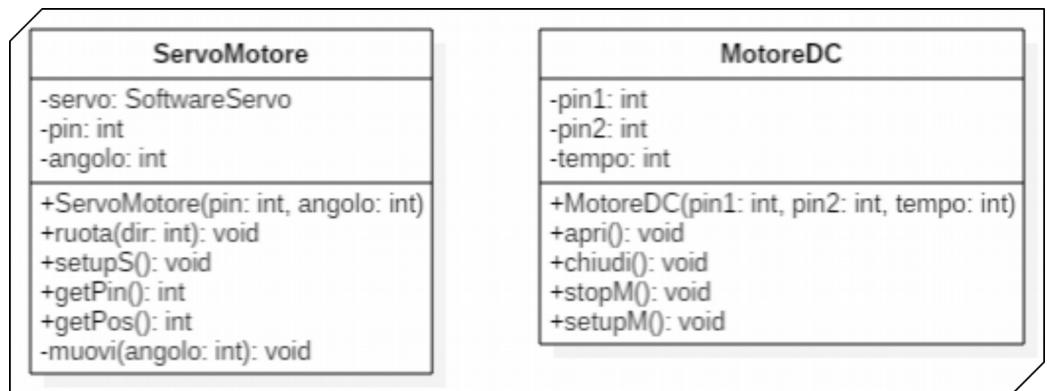


Figura 3 Diagramma delle classi dello Sketch di Arduino.

## Main codes

From CubeParsing. Side acquisition and color detection.

```

44     public int colourOf(int pixel){
45         float[] hsv = new float[3];
46         float h, s, l;
47         Color.colorToHSV(pixel, hsv); //conversione a HSV
48         //conversione da HSV a HSL
49         h = hsv[0];
50         l = (2-hsv[1])*hsv[2];
51         s = hsv[1]*hsv[2];
52         s /= (l<=1) ? l : (2-l);
53         l/=2;
54         l*=100;
55         s*=100;
56         //rilevazione colori
57         if(s<50)
58             if(l<20) return BLACK;
59             else if(l>70) return WHITE;
60         if(l<15) return BLACK;
61         if(l>80) return WHITE;
62         if(45<=h&&h<=72) return YELLOW;
63         if(73<=h&&h<=190) return GREEN;
64         if(200<=h&&h<=260) return BLUE;
65         if(17<=h&&h<=45) return ORANGE;
66         if(h<=16||h>=338) return RED;
67         return UNDEFINED;
68     }

```

Figura 4 Da CubeParsing. Rilevazione colore.

```

71     protected void parseImage(Bitmap img){
72         //valori di coordinate da cui iniziano le caselle
73         int[] inizio = {450, 1050, 1650};
74
75         //rilevazione dei colori delle 9 caselle
76         for(int i=0, inizioX, inizioY; i<9; i++){
77             int[] cont = new int[8];
78             inizioX = inizio[(i%3)];
79             inizioY = inizio[(i/3)];
80             for(int j=0; j<8; j++)
81                 cont[j] = 0;
82
83             //Rilevazione statistica
84             //Il colore rilevato più frequentemente in un'area di 10000px
85             //sarà il colore della casella
86             for(int x=0; x<100; x++)
87                 for(int y=0; y<100; y++)
88                     cont[colourOf(img.getPixel(inizioX+x, inizioY+y))]++;
89             facelet[parsedIndex][i] = MainActivity.massimo(cont);
90         }
91         parsedIndex++; //incremento l'indice della faccia
92         if(parsedIndex<6){
93             //incremento progresso
94             ProgressBar bar = (ProgressBar)callingActivity.findViewById(R.id.progressBar);
95             TextView text = (TextView)callingActivity.findViewById(R.id.loading);
96             bar.setProgress(parsedIndex*10);
97             text.setText("Sto risolvendo il cubo... Acquisizione faccia "+(parsedIndex+1)+".");
98             //chiamata alla funzione per ruotare il cubo alla prossima faccia
99             bluetoothHand.moveHand(parsedIndex);
100        }else
101            parseCube(); //se sono acquisite tutte, analizzo il cubo in generale
102    }

```

Figura 5 Da CubeParsing. Metodo per acquisire una faccia.

In the method **parseCube()**, the matrix facelet is converted into a string and sent to the algorithm classes.

```
94     public void sendToArduino(String command){  
95         /*Invio della stringa command ad Arduino*/  
96         command+=".";  
97         byte[] message = command.getBytes();  
98         try {  
99             output.write(message);  
100        } catch (IOException e) {  
101            Toast.makeText(callingActivity, "Errore nell'invio dei dati via Bluetooth.", Toast.LENGTH_LONG).show();  
102        }  
103        receiveOkFromArduino();  
104    }  
105  
106    //svuota il buffer appena è disponibile  
107    public String receiveFromArduino(){  
108        String result = "";  
109        try{  
110            while(input.available()==0);  
111            if(input.available()>0)  
112                while(input.available()>0){  
113                    result += (char)input.read();  
114                    Thread.sleep(10);  
115                }  
116            }catch(Exception e){ }  
117        return result;  
118    }
```

Figura 6 Da Hand. Metodi per l'invio/ricezione via Bluetooth a/da Arduino.

# Rubik Solver

From the Arduino Sketch:

```
class ServoMotore{
public:
    ServoMotore(int pin, int angoloA, int angoloB, int angolo0){
        this->pin = pin;
        servo.attach(pin);
        this->angoloA = angoloA; //angolo per movimento Antiorario
        this->angoloB = angoloB; //angolo Base
        this->angolo0 = angolo0; //angolo per movimento Orario
        ruota(0); //raggiunge la posizione base
    }
    void ruota(int dir){
        switch(dir){
            case -1: muovi(angoloA); break; //movimento Antiorario
            case 0: muovi(angoloB); break; //Base
            case 1: muovi(angolo0); break; //Orario
            default: Serial.println("Dir sconosciuta in ServoMotore::ruota(). Dir="+dir);
        }
    }
    void muovi(int angolo){ //movimento all'angolo passato per parametro
        int pos = servo.read(), speed = (pos>angolo)?-1:1; //posizione attuale e direzione
        for(bool refresh=false; pos!=angolo; pos+=speed, refresh=!refresh){
            if(refresh) SoftwareServo::refresh(); //refresh del Servo, necessario almeno ogni 20ms
            servo.write(pos);
            delay(10);
        }
    }
}
```

Figura 7 Classe ServoMotore. Metodi principali.

```
class MotoreDC{
public:
    MotoreDC(int pin1, int pin2, int tempo){
        this->pin1 = pin1; //settaggio attributi
        this->pin2 = pin2;
        this->tempo = tempo;
        pinMode(pin1, OUTPUT); //settaggio a OUTPUT dei pin
        pinMode(pin2, OUTPUT);
        setupM(); //posizione aperta
        //per essere sicuri lo si fa aprire per più tempo del dovuto
    }
    void apri(){ //metodo per aprire la mano
        digitalWrite(pin1, HIGH);
        digitalWrite(pin2, LOW);
        delay(tempo);
        stopM();
    }
    void chiudi(){ //metodo per chiudere la mano
        digitalWrite(pin1, LOW);
        digitalWrite(pin2, HIGH);
        delay(tempo);
        stopM();
    }
    void stopM(){ //metodo per fermare il movimento
        digitalWrite(pin1, LOW);
        digitalWrite(pin2, LOW);
    }
}
```

Figura 8 Classe MotoreDC. Metodi principali.