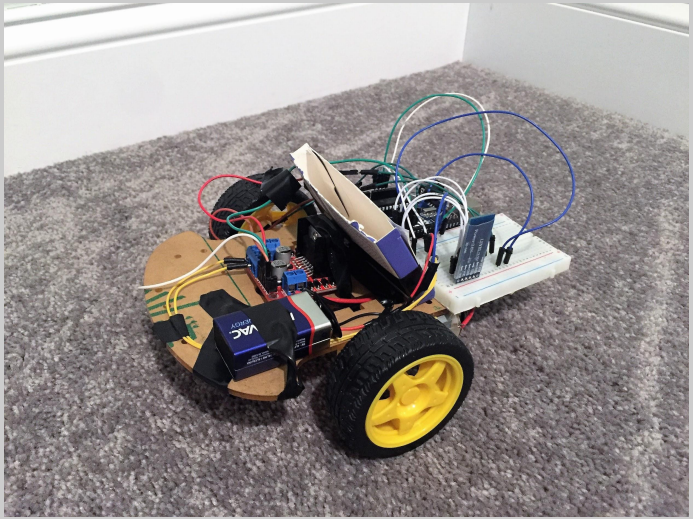
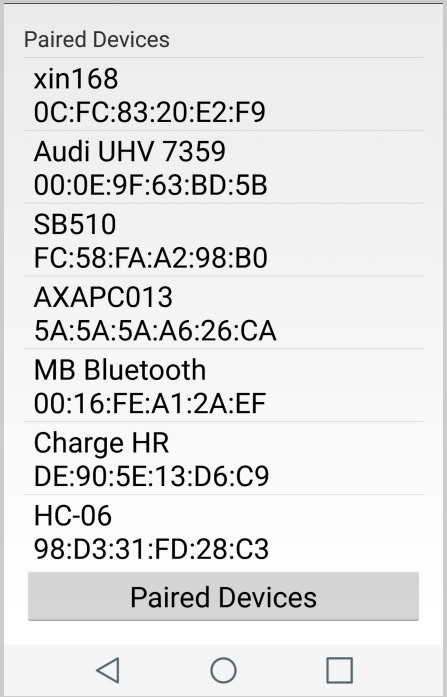
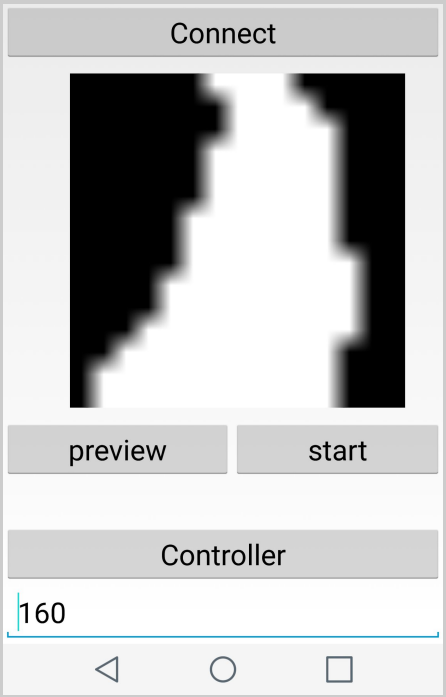
Self-Driving Car

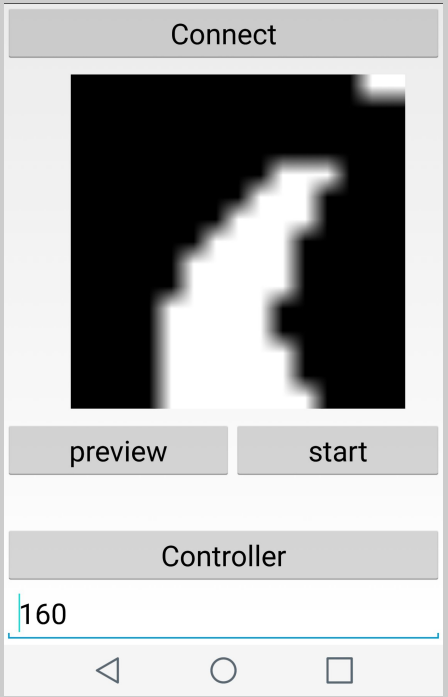


Camden Parsons

Here is the app:







How it works:

-Take a picture with phone camera.

↓

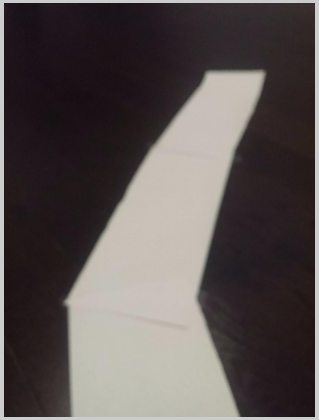
-Convert image into an interpretable form.

↓

-Interpret data from the image and make a decision.

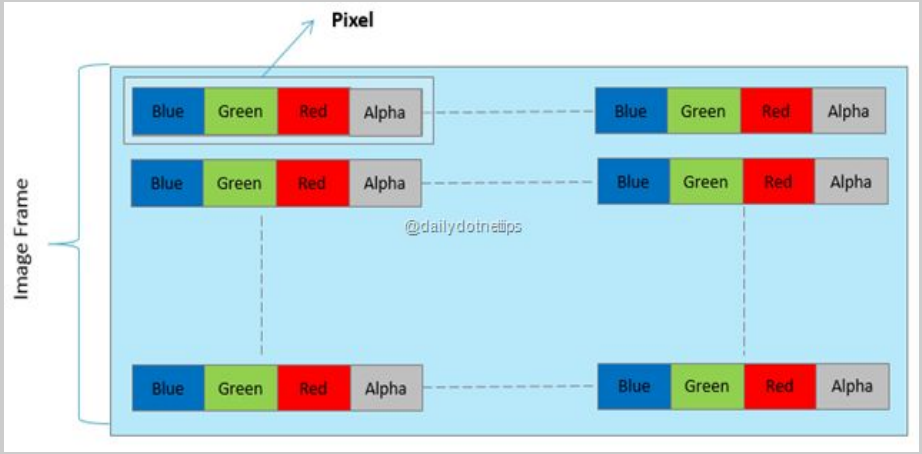
↓

-Send command to rc car via bluetooth



“Turn Right”

“Go Straight”



[0,0,0] , [10,10,40] , [225,180,200] ..... [250,250,250]

↓ average RGB values

0 , 20 , 200 250

↓ compare to threshold

0 , 0 , 255 255

- A picture is just a big array of pixel data
- The group of data that correspond to each pixel in an image represents how bright the red, blue and green light in the pixel should be. (with a number 0-255)
- averaging this group of data shows how white each pixel is.
- I use a threshold specified in the UI of the app to determine what pixels are white enough to be the road

Code for taking and processing image

```
public class CameraView extends Activity implements SurfaceHolder.Callback, OnClickListener{
    private static final String TAG = "CameraTest";
    Camera mCamera;
    boolean mPreviewRunning = false;
    TextView textView;

    /deprecation/
    public void onCreate(Bundle icicle){
        super.onCreate(icicle);
        Log.e(TAG, "onCreate");

        getWindow().setFormat(PixelFormat.TRANSLUCENT);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.cameraview);
        ImageView img = (ImageView) findViewById(R.id.blankImage);

        mSurfaceView = (SurfaceView) findViewById(R.id.surface_camera);
        mSurfaceView.setOnClickListener(this);
        mSurfaceHolder = mSurfaceView.getHolder();
        mSurfaceHolder.addCallback(this);
        mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
       textView = (TextView)findViewById(R.id.textView6);
        textView.setText(CaptureCameraImage.bext_.getText());
    }

    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState){
        super.onRestoreInstanceState(savedInstanceState);
    }


    Camera.PictureCallback mPictureCallback = (data, camera) -> {
            // TODO Auto-generated method stub
            if (data != null){
                //Intent mIntent = new Intent();
                //mIntent.putExtra("image",imageData);

                mCamera.stopPreview();
                mPreviewRunning = false;
                mCamera.release();

                 try{
                     BitmapFactory.Options opts = new BitmapFactory.Options();
                     Bitmap bitmap= BitmapFactory.decodeByteArray(data, 0, data.length,opts);
                     bitmap = Bitmap.createScaledBitmap(bitmap, 15, 15, false);
                     int width = bitmap.getWidth();
                     int height = bitmap.getHeight();

                     // creates matrix for the manipulation
                     Matrix matrix = new Matrix();

                     matrix.postRotate(90);
                     Bitmap resisedBitmap = Bitmap.createBitmap(bitmap, 0, 0,width, height, matrix, true);
                     CaptureCameraImage.image.setImageBitmap(createBlackAndWhite(resisedBitmap));

                 }catch(Exception e){
                     e.printStackTrace();
                 }
                //StoreByteImage(mContext, imageData, 50,"ImageName");
                //setResult(EXTRA_MODE, mIntent);
                setResult(585);
```

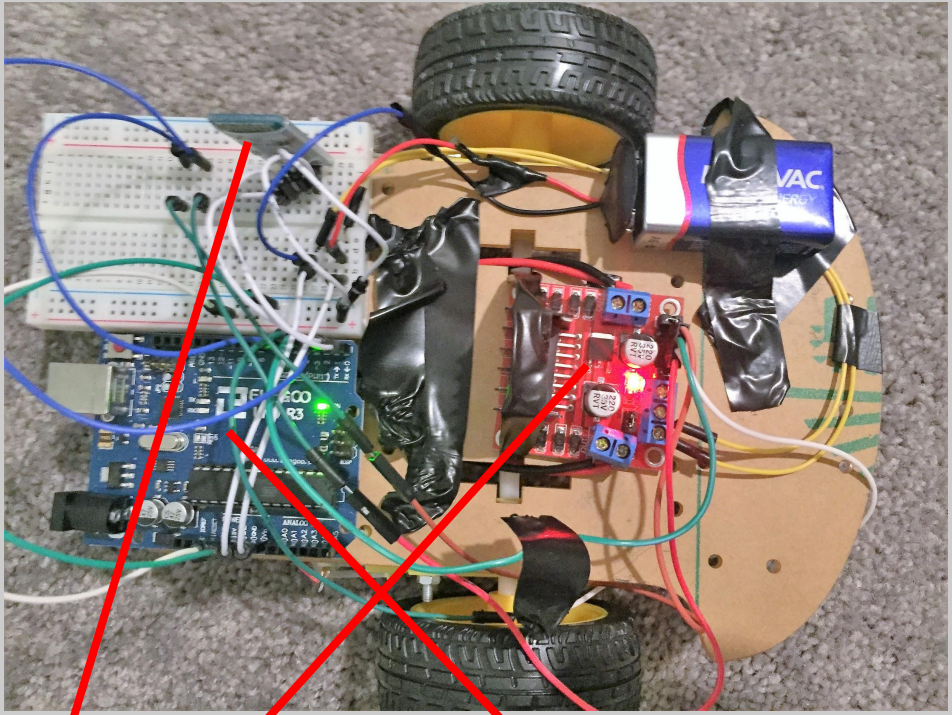
```
    public void stop(View v) { CaptureCameraImage.loop=false; }
    public Bitmap createBlackAndWhite(Bitmap src) {
        int width = src.getWidth();
        int height = src.getHeight();
        // create output bitmap
        Bitmap bmOut = Bitmap.createBitmap(width, height, src.getConfig());
        // color information
        int A, R, G, B;
        int pixel;
        int counter=0;
        int[][] avgs= new int[height][2];
        // scan through all pixels
        for (int x = 0; x < width; ++x) {
            for (int y = 0; y < height; ++y) {
                // get pixel color
                pixel = src.getPixel(x, y);
                A = Color.alpha(pixel);
                R = Color.red(pixel);
                G = Color.green(pixel);
                B = Color.blue(pixel);
                int gray = (int) (0.2989 * R + 0.5870 * G + 0.1140 * B);

                // use 128 as threshold, above -> white, below -> black
                if (gray > CaptureCameraImage.threshold){
                    gray = 255;
                    avgs[y][0]+=(x-width/2);
                    avgs[y][1]+=1;
                    counter++;
                }else
                    gray = 0;
                // set new pixel color to output bitmap
                bmOut.setPixel(x, y, Color.argb(A, gray, gray, gray));
            }
        }
        int sum=0;
        for(int i=0;i<height;i++){
            sum+=avgs[i][0]/(avgs[i][1]+1);
        }

        if((double)counter/(height*width)<.1){
            CaptureCameraImage.bext_.setText("stop");
            ledControl.bt_write("b");
        }else if( sum/(height/2)>=2){
            CaptureCameraImage.bext_.setText("turn right");
            ledControl.bt_write("sr");
        }else if(sum/(height/2)<=-2){
            CaptureCameraImage.bext_.setText("turn left");
            ledControl.bt_write("sl");
        }else{
            CaptureCameraImage.bext_.setText("");
            ledControl.bt_write("s");
        }



        return bmOut;
    }

    protected void onSaveInstanceState(Bundle outState) { super.onSaveInstanceState(outState); }

    protected void onStop(){
        Log.e(TAG, "onStop");
        super.onStop();
```

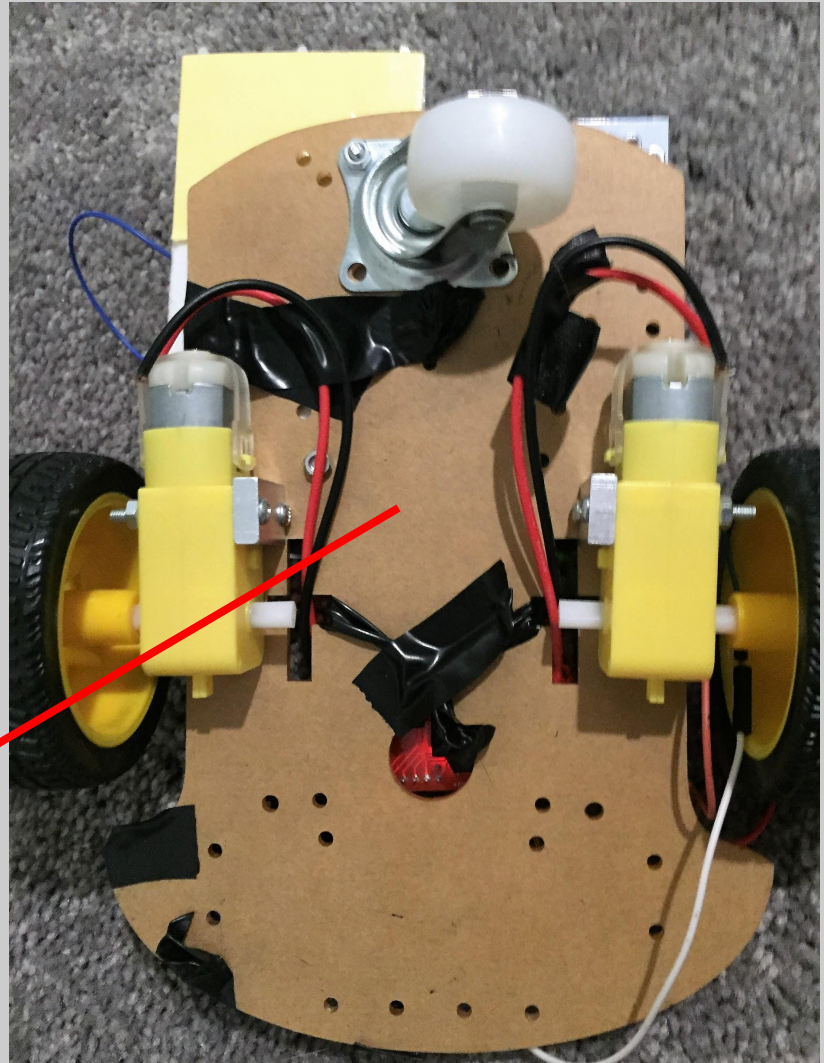
```
    @TargetApi(9)
    public void surfaceCreated(SurfaceHolder holder){
        Log.e(TAG, "surfaceCreated");
        mCamera = Camera.open(CaptureCameraImage.cameraID);
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
        Log.e(TAG, "surfaceChanged");

        // XXX stopPreview() will crash if preview is not running
        if (mPreviewRunning){
            mCamera.stopPreview();
        }

        Camera.Parameters p = mCamera.getParameters();
        p.setPreviewSize(300, 300);

        if(CaptureCameraImage.cameraID == 0){
                    p.setFlashMode(Camera.Parameters.FLASH_MODE_OFF);
        }

        mCamera.setParameters(p);
        try{
            mCamera.setPreviewDisplay(holder);
        }catch (Exception e){
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        mCamera.startPreview();
        mPreviewRunning = true;
        mCamera.takePicture(null, mPictureCallback, mPictureCallback);

        if(CaptureCameraImage.loop){
            try {
                Thread.sleep(350);
                Thread.dumpStack();
            }catch (InterruptedException e){

            };
            Intent i = new Intent(CameraView.this,CameraView.class);
            startActivityForResult(i,999);
        }
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        Log.e(TAG, "surfaceDestroyed");
        //mCamera.stopPreview();
        //mPreviewRunning = false;
        //mCamera.release();
    }
```

Code for connecting and sending message with bluetooth

```
public class bTooth extends Activity {

    Button btnOn, btnOff, btnDis;
    SeekBar brightness;
    TextView lumn;
    String address = null;
    private ProgressDialog progress;
    BluetoothAdapter myBluetooth = null;
    static BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;
    //SPP UUID. Look for it
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        Intent newint = getIntent();
        address = newint.getStringExtra(DeviceList.EXTRA_ADDRESS); //receive the address of the bluetooth device

        //view of the ledControl
        setContentView(R.layout.activity_led_control);

        //call the widgtes
        btnOn = (Button)findViewById(R.id.button2);
        btnOff = (Button)findViewById(R.id.button3);
        btnDis = (Button)findViewById(R.id.button4);
        brightness = (SeekBar)findViewById(R.id.seekBar);
        lumn = (TextView)findViewById(R.id.lumn);

        new ConnectBT().execute(); //Call the class to connect

        //commands to be sent to bluetooth
        btnOn.setOnClickListener((v) -> { turnOnLed();      //method to turn on });

        btnOff.setOnClickListener((v) -> { turnOffLed();   //method to turn off });

        btnDis.setOnClickListener((v) -> { Disconnect(); //close connection });

        brightness.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                if (fromUser==true)
                {
                    lumn.setText(String.valueOf(progress));
                    try
                    {
                        btSocket.getOutputStream().write(String.valueOf(progress).getBytes());
                    }
                    catch (IOException e)
                    {

                    }
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
```

```
    public static void bt_write(String str){
        if (btSocket!=null)
        {
            try
            {
                btSocket.getOutputStream().write(str.toString().getBytes());
            }
            catch (IOException e)
            {
                System.out.println("error");
            }
        }
    }



    // fast way to call Toast
     private void msg(String s)
    {
        Toast.makeText(getApplicationContext(),s,Toast.LENGTH_LONG).show();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        //getMenuInflater().inflate(R.menu.menu_led_control, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        //if (id == R.id.action_settings) {
         //   return true;
        //}

        return super.onOptionsItemSelected(item);
    }

    private class ConnectBT extends AsyncTask<Void, Void, Void>  // UI thread
    {
        private boolean ConnectSuccess = true; //if it's here, it's almost connected

        @Override
        protected void onPreExecute()
        {
            progress = ProgressDialog.show(ledControl.this, "Connecting...", "Please wait!!!");  //show a progress dialog
        }
        @TargetApi(11)
        @Override
        protected Void doInBackground(Void... devices) //while the progress dialog is shown, the connection is done in background
        {
```





Bluetooth module → Arduino

Motor cotroller → motors

LED_Write_Read §

```
char command;
String string;
boolean drive;


  void setup()
  {
    Serial.begin(9600);
    pinMode(13, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(5, OUTPUT);
    drive=false;
  }

  void loop()
  {
    if (Serial.available() > 0)
    {string = "";}

    while(Serial.available() > 0)
    {
      command = ((byte)Serial.read());
      Serial.println(command);
      if(command == ':')
      {
        break;
      }

      else
      {
        string += command;
      }

      delay(1);
    }
```

Code for the arduino to take in bluetooth input and controll motors

```
  if(string=="b"){
    //digitalWrite(5, LOW);
    //digitalWrite(2, LOW);
    drive=false;
  }
  if(string=="sr"){

    digitalWrite(5, HIGH);
    delay(150);
    digitalWrite(5, LOW);
    delay(100);
    drive=false;
    string="";
  }
  if(string=="sl"){
    digitalWrite(2, HIGH);

    delay(150);
    digitalWrite(2, LOW);
    delay(100);
    drive=false;
    string="";
  }
  if(string=="s"){
    //digitalWrite(2, HIGH);
    //digitalWrite(5, HIGH);
    drive=true;
  }

  if(drive){
    for(int l =0;l<2;l++){
      digitalWrite(5, HIGH);
      digitalWrite(2, HIGH);
      delay(50);
      digitalWrite(2, LOW);
      delay(7);

      digitalWrite(5, LOW);
      delay(25);
    }
    string="";
  }else{
    digitalWrite(5, LOW);
    digitalWrite(2, LOW);
  }
```