5.2 PROGRAM

5.2.1 ESP 8266 Code

```
#define BLYNK_TEMPLATE_ID "TMPL3HdHIwHYs"
#define BLYNK_TEMPLATE_NAME "IOT BASED SPING CAR"
#define BLYNK_PRINT Serial
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <SimpleTimer.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Servo.h>
char auth[] = "snqc4ANohS2Au800kikKG9dbmKn2T6BU"; // Blynk auth token
char ssid[] = "SASANGAN"; // Network Name
char pass[] = "sasangandeena"; // Network Password
Servo s1;
Servo s2;
WidgetLCD lcd(V9);
WidgetMap myMap(V100);
String GPSLabel = "BLYNK";
SimpleTimer timer;
static const int RXPin = 13, TXPin = 1;
static const uint32_t GPSBaud = 9600; //if Baud rate 9600 didn't work in your case
then use 4800

TinyGPSPlus gps;                          // The TinyGPS++ object
SoftwareSerial ss(RXPin, TXPin);     // Serial connection to the GPS module
BLYNK_WRITE(V1)
{
  s1.write(param.asInt());
```

```
}
BLYNK_WRITE(V2)
{
  s2.write(param.asInt());
}
BLYNK_WRITE(V3)
{
  int pinval = param.asInt();
  analogWrite(D2,pinval);
}
BLYNK_WRITE(V10)
{
  int pinvak = param.asInt();
  analogWrite(D8,pinvak);
}
BLYNK_WRITE(V4)
{
  int forward = param.asInt();
  digitalWrite(D3,forward);
  digitalWrite(D5,forward);
}
BLYNK_WRITE(V5)
{
  int reverse = param.asInt();
  digitalWrite(D6,reverse);
  digitalWrite(D4,reverse);
}
BLYNK_WRITE(V6)
{
  int right = param.asInt();
  digitalWrite(D5,right);
  digitalWrite(D4,right);
}
BLYNK_WRITE(V7)
{
  int left = param.asInt();
  digitalWrite(D3,left);
  digitalWrite(D6,left);
}
void setup()
{
  pinMode(D3,OUTPUT);   // motor a forward
  pinMode(D4,OUTPUT);   // motor a backward
  pinMode(D5,OUTPUT);   // motor b forward
  pinMode(D6,OUTPUT);   // motor b backward
  Serial.begin(9600);
  ss.begin(GPSBaud);
  Blynk.begin(auth, ssid, pass);
  Serial.println("Activating GPS");
  timer.setInterval(1000L, periodicUpdate);
  timer.setInterval(60*1000, reconnectBlynk);
  s1.attach(16,700,2300);
  s2.attach(5,700,2300);
}
void periodicUpdate() {
  String line1, line2;
  //LCD
  lcd.clear();
  if (gps.location.isValid() && (gps.location.age() < 3000)) {
    //position current
    line1 = String("lat: ") + String(gps.location.lat(), 6);
    line2 = String("lng: ") + String(gps.location.lng(), 6);
    lcd.print(0, 0, line1);
    lcd.print(0, 1, line2);
    //update location on map
    myMap.location(2, gps.location.lat(), gps.location.lng(), GPSLabel);
  } else {
    //position is lost
    lcd.print(0, 0, "GPS lost");
  }
}

void updateGPS() {
  //read data from GPS module
  while (ss.available() > 0) {
  gps.encode(ss.read());
  }
}

void reconnectBlynk() {
  if (!Blynk.connected()) {
    Serial.println("Lost connection");
    if(Blynk.connect()) Serial.println("Reconnected");
    else Serial.println("Not reconnected");
  }
}

void loop()
{
  timer.run();
  Blynk.run();
  updateGPS();
```

```
}
```

5.2.2 ESP 32 Camera Module Code

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//
//            You must select partition scheme from the board menu that has at least 3MB
APP space.
//            Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes
up from 15
//            seconds to process single frame. Face Detection is ENABLED if PSRAM is
enabled as well

PSRAM
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char* ssid = "SASANGAN";
const char* password = "sasangandeena";

void startCameraServer();
void setupLedFlash(int pin);
void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                      for larger pre-allocated frame buffer.
  if(config.pixel_format == PIXFORMAT_JPEG){
  if(psramFound()){
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```