

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | دوربین کنترل از راه دور با موبایل | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | سینا حسنی |  | | | |
|  | | | | —مبانی رباتیک—استاد: محمد زارعبهمن 1402 |  | | | |
|  | | |  | | |  | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | | |  |
|  | چکیده | | | | | | |  |
|  |  | | |  |  | | |  |
|  |  | |  | | |  | |  |
|  |  |  | این پروژه شامل طراحی و ساخت یک ربات، پیکربندی‌های پیشرفته لینوکس در رزبری پای، و ساخت یک برنامه اندرویدی برای کنترل ربات است. ابتدا ربات با استفاده از پلکسی‌گلاس، ورق‌های پلاستیکی، موتورهای DC و قطعات الکترونیکی ساخته می‌شود و قادر به حرکت و استفاده از چراغ جلو خواهد بود. سپس رزبری پای را پیکربندی کرده و وابستگی‌های مورد نیاز نصب می‌شود. در نهایت، یک برنامه اندرویدی ساخته می‌شود که با استفاده از دوربین و اتصال وای‌فای، ربات را از راه دور کنترل می‌کند. | | |  |  |  |
|  | | | | |
|  |  |  |  |
|  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Decorative | |  |  |  |  | |
|  | | THE PROCESS | | |  | |
|  | |  |  |  |  | |
|  | مقدمه این پروژه یک دوربین کنترل از راه دور است که با استفاده از مینی کامپیوتر رزبری‌پای 3 برای اتصال به شبکه و پردازش تصاویر استفاده شده است و از میکروکنترلر آردوینو برای کنترل موتور و سنسورها استفاده شده. بدنه ربات از ورق پلاکسی‌گلاس و ورق پلاستیک می‌باشد. از تایر و موتور DC برای حرکت به اطراف استفاده شده. از دوربین رزبری‌پای برای ثبت تصاویر و ویدیو استفاده می‌شود. از انواع قطعات مختاف الکترونیکی مثل ترانزیستور و مقاومت برای کنترل بخش‌های ربات استفاده شده است. و در نهایت با استفاده از اپلیکیشن بر روی تلفن همراه میتوان تصاویر را دریافت کرد و آن را کنترل کرد.  این پروژه می‌تواند برای کاربردهای مختلفی مانند نظارت خانگی، سرگرمی و تحقیقات علمی استفاده شود. به عنوان مثال، می‌توان از آن برای استفاده در مسیر های کوچک و باریک استفاده کرد. همچنین میتوان به عنوان اسباب‌بازی که با تلفن همراه کنترل می‌شود نیز استفاده کرد. | | | | |  |

|  |  |  |
| --- | --- | --- |
|  | روش کار  1. **مواد و قطعات مورد نیاز**:    * میکروکنترلر Arduino pro mini 328p    * مینی کامپیوتر Rasbpberry Pi 3    * قاب برای Rasbpberry Pi    * دوربین Rasbpberry Pi    * ورق پلکسی‌گلاس    * ورق پلاستیکی    * LED برای چراغ    * برد PCB    * سنسور موانع مادون قرمز    * رگولاتور L7805CV پنچ ولت    * مقاومت 200 اهم    * باتری 5 ولت    * سوییچ خاموش و روشن    * Arduino IDE 2. **مراحل اجرا**:    * ربات از پلکسی‌گلاس یا پلاستیک سخت ساخته می‌شود و صفحه پایه 18 در 13 سانتی‌متر خواهد بود. موتورهای DC با براکت‌های فلزی به صفحه پایه متصل می‌شوند و پل H در وسط صفحه نصب می‌شود. چرخ‌های عقب با فاصله‌دهنده‌های فلزی شش‌ضلعی 2 سانتی‌متری متصل می‌شوند. یک سوراخ بزرگ نزدیک پل H برای اتصال قطعات الکترونیکی در قسمت بالای صفحه لازم است. قسمت بالایی ربات شامل دو صفحه به شکل "L" خواهد بود که به هم چسبانده شده و پوششی برای قطعات الکترونیکی و پایه‌ای برای کیس رزبری پای فراهم می‌کند. این قسمت با فاصله‌دهنده‌های فلزی شش‌ضلعی 6 سانتی‌متری به قسمت پایینی متصل می‌شود.    * سپس، پل H در کف ربات نصب شده و دو سنسور مادون قرمز در جلو و عقب شاسی با استفاده از صفحه فلزی "L" شکل متصل می‌شوند. سپس، چراغ جلو LED 5 ولت در وسط قسمت بالایی ربات نصب شده و کابل‌ها از طریق سوراخی به یک کانکتور دو سیم ماده لحیم می‌شوند. در نهایت، کیس رزبری پای با دوربین، کارت حافظه 4 گیگابایتی و کانکتور دوربین مونتاژ می‌شود و کارت حافظه با آخرین نسخه Raspian نصب شده وارد می‌شود.    * در مرحله بعد، اپلیکیشن اندروید تصویر زنده از دوربین رزبری پای را نمایش می‌دهد و دستورات روشنایی و موتور را به سرور MQTT ارسال می‌کند. سرور پایتون در رزبری پای این دستورات را به آردوینو ارسال کرده و آردوینو نیز فواصل سنسورها را به سرور پایتون برمی‌گرداند. داده‌ها از طریق MQTT به اپلیکیشن اندروید منتقل شده و نمایش داده می‌شوند. برای شروع، Raspbian باید نصب و پیکربندی شود و دوربین به طور فیزیکی متصل و تنظیم گردد.    * در مرحله بعدی برنامه اندرویدی را برای ارتباط با ربات ساخته و پیکربندی میکنیم.    * آردوینو دستورات موتور و نور از طریق خط سریال دریافت می‌کند و موتورها را حرکت می‌دهد یا نور را روشن یا خاموش می‌کند و سنسورهای مادون قرمز از جلو و عقب ربات را بررسی می‌کند و داده‌های مربوط به فواصل را از طریق خط سریال ارسال می‌کند. 3. **کد آردوینو**:   // source for TextMotorCommandsInterpretter: "https://github.com/danionescu0/arduino/tree/master/libraries/TextMotorCommandsInterpretter"  #include <SoftwareSerial.h>  #include <TextMotorCommandsInterpretter.h>  const char MOTOR\_COMMAND = 'M';  const char LIGHT\_COMMAND = 'L';  /\*\*  \* how long the motor command will take effect in ms  \* an incomming motor command will last for maxDurationForMottorCommand  \* if it's not going to be resetted by another motor command  \*/  const long maxDurationForMottorCommand = 300;  // adjust this value to limit robot speed  const byte maxPwmValue = 230;  // How long between successive distance transmissions in ms  const long transmitingInterval = 500;  const int maxObstacleDetection = 1000; // analog read max detection value  const int minObstacleDetection = 500; // analog read min detection value  const byte FLASH\_PIN = 3;  const byte RIGHT\_MOTOR\_PWM\_PIN = 5;  const byte RIGHT\_MOTOR\_EN1\_PIN = A4;  const byte RIGHT\_MOTOR\_EN2\_PIN = A5;  const byte LEFT\_MOTOR\_PWM\_PIN = 6;  const byte LEFT\_MOTOR\_EN1\_PIN = A3;  const byte LEFT\_MOTOR\_EN2\_PIN = A2;  const byte FRONT\_DISTANCE\_SENSOR = A0;  const byte BACK\_DISTANCE\_SENSOR = A1;  SoftwareSerial masterComm(11, 10); // RX, TX  TextMotorCommandsInterpretter motorCommandsInterpretter(-50, 50, -50, 50);  String currentCommand;  long lastCheckedTime;  long lastTransmitTime;  boolean inMotion = false;  void setup()  {  Serial.begin(9600);  masterComm.begin(9600);  masterComm.setTimeout(10);  pinMode(FLASH\_PIN, OUTPUT);  pinMode(LEFT\_MOTOR\_PWM\_PIN, OUTPUT);  pinMode(LEFT\_MOTOR\_EN1\_PIN, OUTPUT);  pinMode(LEFT\_MOTOR\_EN2\_PIN, OUTPUT);  pinMode(RIGHT\_MOTOR\_PWM\_PIN, OUTPUT);  pinMode(RIGHT\_MOTOR\_EN1\_PIN, OUTPUT);  pinMode(RIGHT\_MOTOR\_EN2\_PIN, OUTPUT);  lastCheckedTime = millis();  lastTransmitTime = millis();  }  void loop()  {  if (masterComm.available() > 0) {  currentCommand = masterComm.readString();  processCommand();  }  if (inMotion && millis() - lastCheckedTime > maxDurationForMottorCommand) {  stopMotors();  }  if (millis() - lastTransmitTime > transmitingInterval) {  lastTransmitTime = millis();  masterComm.print(getObstacleData());  Serial.print(analogRead(BACK\_DISTANCE\_SENSOR));Serial.print("---");  Serial.println(getObstacleData());  }  /\* FOR DEBUG  motorCommandsInterpretter.analizeText("M:-14:40;");  Serial.write("Left==");Serial.println(motorCommandsInterpretter.getPercentLeft());  Serial.write("Right==");Serial.println(motorCommandsInterpretter.getPercentRight());  delay(10000);\*/  }  String getObstacleData()  {  int frontDistance = analogRead(FRONT\_DISTANCE\_SENSOR);  int backDistace = analogRead(BACK\_DISTANCE\_SENSOR);  frontDistance = map(frontDistance, maxObstacleDetection, minObstacleDetection, 0, 10);  backDistace = map(backDistace, maxObstacleDetection, minObstacleDetection, 0, 10);  return String("F=" + String(frontDistance) + ":B=" + String(backDistace) + ";");  }  void processCommand()  {  switch (currentCommand.charAt(0)) {  case (MOTOR\_COMMAND):  steerCar();  break;  case (LIGHT\_COMMAND):  toggleLight(currentCommand.charAt(2));  break;  }  }  void steerCar()  {  motorCommandsInterpretter.analizeText(currentCommand);  float percentLeftMotor = motorCommandsInterpretter.getPercentLeft();  float percentRightMotor = motorCommandsInterpretter.getPercentRight();  Serial.write("Left=");Serial.println(percentLeftMotor);  Serial.write("Right=");Serial.println(percentRightMotor);  setMotorsDirection(motorCommandsInterpretter.getDirection());  analogWrite(LEFT\_MOTOR\_PWM\_PIN, percentLeftMotor \* maxPwmValue);  analogWrite(RIGHT\_MOTOR\_PWM\_PIN, percentRightMotor \* maxPwmValue);  inMotion = true;  lastCheckedTime = millis();  }  void setMotorsDirection(boolean forward)  {  if (forward) {  digitalWrite(LEFT\_MOTOR\_EN1\_PIN, HIGH);  digitalWrite(LEFT\_MOTOR\_EN2\_PIN, LOW);  digitalWrite(RIGHT\_MOTOR\_EN1\_PIN, HIGH);  digitalWrite(RIGHT\_MOTOR\_EN2\_PIN, LOW);  } else {  digitalWrite(LEFT\_MOTOR\_EN1\_PIN, LOW);  digitalWrite(LEFT\_MOTOR\_EN2\_PIN, HIGH);  digitalWrite(RIGHT\_MOTOR\_EN1\_PIN, LOW);  digitalWrite(RIGHT\_MOTOR\_EN2\_PIN, HIGH);  }  }  void stopMotors()  {  Serial.println("Stopping motors");  analogWrite(LEFT\_MOTOR\_PWM\_PIN, 0);  analogWrite(RIGHT\_MOTOR\_PWM\_PIN, 0);  inMotion = false;  }  void toggleLight(char command)  {  Serial.println("Toggle light");  if (command == '1') {  digitalWrite(FLASH\_PIN, HIGH);  } else {  digitalWrite(FLASH\_PIN, LOW);  }  }  توضیحات توابع:   * setup(): در این تابع، پین‌ها و کانال‌های سریال برای ارتباط با ماژول خارجی (مثلاً بلوتوث) تنظیم می‌شوند. همچنین پارامترهای مربوط به ارتباط سریال و پین‌های خروجی برای موتورها و سنسورها اعلام و تنظیم می‌شوند. * loop(): این تابع به طور مداوم اجرا می‌شود و از ورودی سریال داده‌های جدید را می‌خواند. اگر دستوری دریافت شود، تابع processCommand() فراخوانی می‌شود تا دستور مربوطه را اجرا کند. سپس، وضعیت موتورها و سنسورها بررسی و اطلاعات مورد نیاز از طریق خط سریال ارسال می‌شوند. * getObstacleData(): این تابع مقادیر خوانده شده از سنسورهای مادون قرمز جلو و عقب ربات را خوانده، آن‌ها را به دسته‌ای دیگر تبدیل کرده و به صورت یک رشته ارسال می‌کند که شامل فاصله‌های محاسبه شده است. * :processCommand()این تابع دستور دریافتی را بررسی می‌کند و به تابع مناسب steerCar() یا toggleLight() تحویل می‌دهد برای اجرای عملیات مربوطه. * steerCar(): در این تابع، دستورات موتور از طریق خط سریال دریافت شده و تفسیر می‌شوند. سرعت موتورها بر اساس دستور دریافتی تنظیم می‌شود و جهت حرکت موتورها تعیین می‌شود. همچنین زمانی که آخرین دستور اجرا شده است و کنترل موتورها قطع شده است، محاسبه می‌شود. * setMotorsDirection(): این تابع جهت حرکت موتورها را بر اساس ورودی مشخص می‌کند (جلو یا عقب). * stopMotors(): این تابع تمامی موتورها را متوقف می‌کند. * :toggleLight() این تابع چراغ روشنایی ربات را بر اساس دستور دریافتی روشن یا خاموش می‌کند. |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | نتیجه گیری ا[ین پروژه جالب تمام چیزهای مورد نیاز برای ساخت یک دوربین نظارت از راه دور قابل حمل را پوشش می‌دهد](https://www.hackster.io/danionescu/mobile-remote-surveillance-camera-35519d" \t "_blank). در این پروژه، از پلکسی‌گلاس، ورق‌های پلاستیکی، موتورهای DC با گیربکس و اجزای الکترونیکی مختلف برای ساخت ربات استفاده می‌شود. این دستگاه قادر به حرکت مستقل دو چرخ جلویی خود است و می‌تواند از چراغ جلویی خود استفاده کند. سپس رزبری پای را برای تغذیه ربات پیکربندی کرده و پروژه را پیکربندی و وابستگی‌های مختلف را نصب می‌کنیم. در نهایت، یک اپلیکیشن اندروید را ساخته و نصب می‌کنیم و از طریق دوربین و اتصال وای‌فای آن را به‌صورت از راه دور کنترل می‌کنیم. [این پروژه مفاهیم و تکنولوژی‌های مختلفی را شامل می‌شود، از جمله پلتفرم‌های توسعه مانند آردوینو، رزبری پای، اپلیکیشن‌های اندروید و الکترونیک، استفاده از H-bridge و ترانزیستور و لینوکس](https://projecthub.arduino.cc/danionescu/e61a8c91-c1b8-4041-9ba1-d1874058b3eb). منابعMobile Remote Surveillance Camera <https://www.hackster.io/danionescu/mobile-remote-surveillance-camera-35519d>  [Mobile Remote Surveillance Camera | Arduino Project Hub](https://projecthub.arduino.cc/danionescu/mobile-remote-surveillance-camera-e61a8c) Android application <https://github.com/danionescu0/android-robot-camera> Main repository for arduino, python code <https://github.com/danionescu0/robot-camera-platform>  <https://chatgpt.com>  <https://copilot.microsoft.com> |  | |
|  |  | |  |
|  |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |