Scenario

Joshua’s friend Pavin needs to get something from Joshua’s house. He knows that he can unlock Joshua’s front door by moving his car to different spots when parking his car in Joshua’s driveway. Help Pavin solve this driveway puzzle.

Pavin knows the following:

The lock has four stages.

He must trigger each stage by moving his car to different positions on the driveway.

He must wait at each position for more than 1 second before moving again.

He can get clues by looking at the 4 green lights hidden is Joshua’s front garden.

Pavin also knows that the code for Joshua’s Arduino controller is uploaded to the GotHub repository.

Your solution

1. List the main steps necessary to unlock the front door.
2. List the devices and pin assignments for all sensors and LEDs used in Joshua’s application.
3. Sketch a rough situation and wiring diagram for Joshua’s application.
4. List the setup initialization steps that are used in Joshua’s application.

Arduino Driveway Application Code

/\*\*/#include <NewPing.h>#define TRIGGER\_PIN 6 // Arduino pin tied to trigger pin on the ultrasonic sensor.#define ECHO\_PIN 7 // Arduino pin tied to echo pin on the ultrasonic sensor.#define MAX\_DISTANCE 200 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.int STAGE1\_LED\_PIN = 8; // Green LEDint STAGE2\_LED\_PIN = 9; // Green LEDint STAGE3\_LED\_PIN = 10; // Green LEDint STAGE4\_LED\_PIN = 11; // Green LEDint DOOR\_LED\_PIN = 12; // Red LED, On is Liocked, Off is Unlocked;int PHOTOCELL\_PIN = A0; // Analog Pin 0boolean isSetStage1 = false;boolean isSetStage2 = false;boolean isSetStage3 = false;boolean isSetStage4 = false;NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE); // NewPing setup of pins and maximum distance.void setup() { initializeConsole();\ initializeApplication(); Serial.println("DEBUG: Initialization Completed."); }void initializeApplication() { initializeLED(STAGE1\_LED\_PIN); initializeLED(STAGE2\_LED\_PIN); initializeLED(STAGE3\_LED\_PIN); initializeLED(STAGE4\_LED\_PIN); initializeLED(DOOR\_LED\_PIN); turnOnLED(DOOR\_LED\_PIN); isSetStage1 = false; isSetStage2 = false; isSetStage3 = false; isSetStage4 = false; }void loop() { if (isSetStage1 == false) { int lightLevel = readPhotocell(PHOTOCELL\_PIN); Serial.print("DEBUG: Photocell Light Level = "); Serial.println(lightLevel); if (lightLevel < 300) { Serial.println("DEBUG: Stage 1 Set."); isSetStage1 = true; blinkTheLED(STAGE1\_LED\_PIN,2,250); turnOnLED(STAGE1\_LED\_PIN); } } if (isSetStage1 == false) { delay(100); return; } if (isSetStage2 == false) { int measuredDistance = readUltrasonic(); if ((measuredDistance >= 15) && (measuredDistance <= 20)) { Serial.print("DEBUG: Ultrasonic Measured Distance 2 = "); Serial.println(measuredDistance); delay(1000); if ((measuredDistance >= 15) && (measuredDistance <= 20)) { Serial.println("DEBUG: Stage 2 Set."); isSetStage2 = true; blinkTheLED(STAGE2\_LED\_PIN,2,250); turnOnLED(STAGE2\_LED\_PIN); } } } if (isSetStage2 == false) { delay(100); return; } if (isSetStage3 == false) { int measuredDistance = readUltrasonic(); if ((measuredDistance >= 5) && (measuredDistance <= 10)) { Serial.print("DEBUG: Ultrasonic Measured Distance 3 = "); Serial.println(measuredDistance); delay(1000); if ((measuredDistance >= 5) && (measuredDistance <= 10)) { Serial.println("DEBUG: Stage 3 Set."); isSetStage3 = true; blinkTheLED(STAGE3\_LED\_PIN,2,250); turnOnLED(STAGE3\_LED\_PIN); } } } if (isSetStage3 == false) { delay(100); return; } if (isSetStage4 == false) { int measuredDistance = readUltrasonic(); if ((measuredDistance >= 15) && (measuredDistance <= 20)) { Serial.print("DEBUG: Ultrasonic Measured Distance = "); Serial.println(measuredDistance); delay(1000); if ((measuredDistance >= 15) && (measuredDistance <= 20)) { Serial.println("DEBUG: Stage 4 Set."); isSetStage4 = true; blinkTheLED(STAGE4\_LED\_PIN,2,250); turnOnLED(STAGE4\_LED\_PIN); } } } if (isSetStage4 == false) { delay(100); return; } if (isSetStage4 == true) { Serial.println("DEBUG: Door Successfully Unlocked!!!"); delay(1000); turnOffLED(DOOR\_LED\_PIN); String doReset = readAString (); initializeApplication(); }}/\* \* This function initializes and external LED for use. \* \* Parameters: \* - ledPin is the number of the Arduino pin that the external LED is connected to \* \* Returns: \* - Nothing \* \* Usage in Setup block \* int myLedPin = 11; // Note: Change this value for your configuration \* initializeLED(myLedPin) \* \*/void initializeLED (int ledPin) { pinMode(ledPin, OUTPUT); digitalWrite(ledPin, LOW);} /\* \* This function a LED a specified number of times with a specified \* delay between blinks. \* \* Parameters: \* - ledPin is the number of the Arduino pin that the external LED is connected to \* - blinkTimes is the number of times to blink the LED \* - delayTime is the delay between turning the LED on and off \* (the amount of time the LED stays lit) \* \* Returns: \* - Nothing \* \* Usage in ther main loop: \* int myLedPin = 11; // Note: Change this value for your configuration \* int timesToBlink = 3: // Blink 3 times. Change it to what you need \* int stayLitTime = 500; // Stay lit 0.5 seconds for each blink \* blinkTheLED(myLedPin, timesToBlink, stayLitTime); \* \*/void blinkTheLED (int ledPin, int blinkTimes, int delayTime) { for (int i = 1; i <= blinkTimes; i++) { digitalWrite(ledPin, HIGH); delay(delayTime); digitalWrite(ledPin, LOW); delay(delayTime); }}void turnOnLED (int ledPin) { digitalWrite(ledPin, HIGH);}void turnOffLED (int ledPin) { digitalWrite(ledPin, LOW);}/\* \* This function initializes the Serial console for reading and writing. \* \* Parameters: \* - None \* \* Returns: \* - Nothing \* \* Usage in Setup block \* initializeConsole(); \* \*/void initializeConsole () { Serial.begin(9600);} /\* \* This function reads a number from the Serial console. \* \* Parameters: \* - None \* \* Returns: \* - The number typed by a used into the serial console. \* \* Usage in ther main loop: \* int myNumber = 0; \* myNumber = readANumber(); \* \*/int readANumber () { Serial.println("Please enter a number."); // Keep waiting for the user to type something while(Serial.available() == 0) { } // Read what was typed int theNumber = Serial.parseInt(); return theNumber;}/\* \* This function reads a string from the Serial console. \* \* Parameters: \* - None \* \* Returns: \* - The string typed by a used into the serial console. \* \* Usage in ther main loop: \* String myCommand = "None"; \* myCommand = readAString(); \* \*/String readAString () { Serial.println("Please enter a string."); // Keep waiting for the user to type something while(Serial.available() == 0) { } // Read what was typed String theString = Serial.readString(); return theString;}/\* \* Initialization. \* \* The Photocell is connected to an Analog input pin and \* does not require initialization. \* \*/ /\* \* This function reads a number from the Photocell. \* The number returned depends on the light level measured \* as follows: \* lowest light level is 200 \* highest light level is 800 \* medium light levels return between 200 to 800 \* The above constraints may be adjusted as needed \* \* Parameters: \* - photocellPin is the number of analog pin that the photocell is connected to \* \* Returns: \* - A number between 200 to 800 corresponding to the light level measured. \* \* Usage in ther main loop: \* int analogPin = 5; \* int lightLevel = 0; \* lightLevel = readPhotocell(analogPin); \* \*/int readPhotocell (int photocellPin) { int photocellValue = 0; photocellValue = analogRead(photocellPin); //adjust the constraint depending on lighting environment. photocellValue = constrain(photocellValue, 200, 800); return photocellValue;}/\* \* This function reads a distance from the Ultrasonic Sensor. \* \* Parameters: \* - none \* \* Returns: \* - A distance to the object measured in cm. \* \* Usage in ther main loop: \* int measuredDistance = 0; \* measuredDistance = readUltrasonic(); \* \*/int readUltrasonic () { delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings. unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS). int distanceMeasured = (uS / US\_ROUNDTRIP\_CM); // Convert ping time to distance in cm and print result (0 = outside set distance range) return distanceMeasured;}