|  |
| --- |
| // Built-in libraries |
|  | #include <Stepper.h> |
|  | #include <binary.h> |
|  | #include <EEPROM.h> |
|  |  |
|  | // 3rd-party libraries |
|  | #include <LedControlMS.h> |
|  | #include <SevenSeg.h> |
|  |  |
|  | // Set up GPIO pins and hardware libraries |
|  | int upButtons[] = {22, 23, 24, 25}; |
|  | int upLights[] = {26, 27, 28, 29}; |
|  | int downButtons[] = {30, 31, 32, 33}; |
|  | int downLights[] = {34, 35, 36, 37}; |
|  |  |
|  | int floorButtons[] = {44, 45, 46, 47, 48}; |
|  | int floorLights[] = {49, 50, 51, 52, 53}; |
|  |  |
|  | Stepper mainMotor(4096, 9, 10, 11, 12); |
|  | SevenSeg floorDisplay(2, 3, 4, 5, 6, 7, 8); // Instantiate a seven segment controller object |
|  | LedControl directionMatrix = LedControl(39, 43, 41, 1); // (from left to right from the back: red/blk/yellow/brown/green) |
|  |  |
|  | // Logic constants |
|  | const int FLOORS = 5; |
|  | const int FLOOR\_TIMER\_START = 30000; // Number of loops to pause at a floor |
|  | const int STEPS\_PER\_TICK = 100; |
|  |  |
|  | const int UP = 1; |
|  | const int IDLING = 0; |
|  | const int DOWN = -1; |
|  | const int BOTH = 2; // In case some floor requests both directions |
|  | const int EITHER = 3; // In case no call requests, but need to go there from inside |
|  |  |
|  | // Logic variables |
|  | int floorRequests[] = {IDLING, IDLING, IDLING, IDLING, IDLING}; |
|  |  |
|  | int currentDirection = IDLING; |
|  | int destinationFloor = -1; |
|  | int floorTimer = 0; |
|  |  |
|  | struct PersistentData { |
|  | int floorLevels[FLOORS]; |
|  | int currentPosition = 0; // (measured in steps) |
|  | int currentFloor = 0; |
|  | } persistentData; |
|  |  |
|  | void setup() { |
|  | Serial.begin(9600); |
|  |  |
|  | mainMotor.setSpeed(4); |
|  |  |
|  | for (int i = 0; i < FLOORS - 1; i++) { |
|  | pinMode(upButtons[i], INPUT\_PULLUP); |
|  | pinMode(downButtons[i], INPUT\_PULLUP); |
|  | pinMode(upLights[i], OUTPUT); |
|  | pinMode(downLights[i], OUTPUT); |
|  | } |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | pinMode(floorButtons[i], INPUT\_PULLUP); |
|  | pinMode(floorLights[i], OUTPUT); |
|  | } |
|  |  |
|  | floorDisplay.setCommonCathode(); |
|  |  |
|  | // Wake up the LED matrix from power-saving mode |
|  | directionMatrix.shutdown(0, false); |
|  | // Set a medium brightness for the matrix |
|  | directionMatrix.setIntensity(0, 8); |
|  |  |
|  | EEPROM.get(0, persistentData); |
|  | floorDisplay.writeDigit(persistentData.currentFloor + 1); |
|  |  |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | Serial.println("Loaded data:"); |
|  | Serial.print("Floor "); |
|  | Serial.print(i + 1); |
|  | Serial.print(": "); |
|  | Serial.println(persistentData.floorLevels[i]); |
|  | } |
|  | Serial.print("Motor at: "); |
|  | Serial.println(persistentData.currentPosition); |
|  |  |
|  | selfTest(); |
|  | } |
|  |  |
|  | void selfTest() { |
|  | for (int i = 0; i < FLOORS - 1; i++) { |
|  | Serial.print("Up "); |
|  | Serial.println(i); |
|  | quickLight(upLights[i]); |
|  | Serial.print("Down "); |
|  | Serial.println(i); |
|  | quickLight(downLights[i]); |
|  | } |
|  |  |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | Serial.print("Floor "); |
|  | Serial.println(i); |
|  | quickLight(floorLights[i]); |
|  | } |
|  |  |
|  | for (int i = 0; i <= 9; i++) { |
|  | Serial.print("Setting 7-segment to "); |
|  | Serial.println(i); |
|  | floorDisplay.writeDigit(i); |
|  | delay(200); |
|  | } |
|  | floorDisplay.writeDigit(persistentData.currentFloor + 1); |
|  |  |
|  | Serial.println("Up arrow"); |
|  | showUpArrow(); |
|  | delay(400); |
|  | Serial.println("Down arrow"); |
|  | showDownArrow(); |
|  | delay(400); |
|  | Serial.println("Clear arrows"); |
|  | turnOffArrows(); |
|  | } |
|  |  |
|  | void quickLight(int pin) { |
|  | digitalWrite(pin, HIGH); |
|  | delay(200); |
|  | digitalWrite(pin, LOW); |
|  | delay(200); |
|  | } |
|  |  |
|  | void loop() { |
|  | // If we're pausing at a floor, blink the corners of the display |
|  | if (floorTimer != 0) { |
|  | if ((floorTimer % 1000) == 0) { |
|  | setMatrixCorners((floorTimer % 2000) == 0); |
|  | } |
|  | floorTimer--; |
|  | } |
|  |  |
|  | // If any call buttons have been pressed, register that and light their lights |
|  | for (int i = 0; i < FLOORS - 1; i++) { |
|  | if (digitalRead(upButtons[i]) == HIGH) { |
|  | if (floorRequests[i] == IDLING || floorRequests[i] == UP) { |
|  | floorRequests[i] = UP; |
|  | } else { |
|  | floorRequests[i] = BOTH; |
|  | } |
|  | digitalWrite(upLights[i], HIGH); |
|  | calcDestination(); |
|  | printStatus("UP PRESSED"); |
|  | } |
|  | if (digitalRead(downButtons[i]) == HIGH) { |
|  | if (floorRequests[i + 1] == IDLING || floorRequests[i + 1] == DOWN) { |
|  | floorRequests[i + 1] = DOWN; |
|  | } else { |
|  | floorRequests[i + 1] = BOTH; |
|  | } |
|  | digitalWrite(downLights[i], HIGH); |
|  | calcDestination(); |
|  | printStatus("DOWN PRESSED"); |
|  | } |
|  | } |
|  |  |
|  | // If any inside floor buttons have been pressed, register that and light their lights |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | if (digitalRead(floorButtons[i]) == HIGH) { |
|  | if (i == 0 && digitalRead(floorButtons[4]) == HIGH) { |
|  | // EASTER EGG! Calibration mode. |
|  | enterCalibrationMode(); |
|  | } else { |
|  | if (floorRequests[i] == IDLING) { |
|  | floorRequests[i] = EITHER; |
|  | } |
|  | digitalWrite(floorLights[i], HIGH); |
|  | } |
|  | calcDestination(); |
|  | printStatus("FLOOR PRESSED"); |
|  | } |
|  | } |
|  |  |
|  | // Go towards our destination |
|  | if (floorTimer == 0 && currentDirection != IDLING) { |
|  | if (currentDirection == UP) { |
|  | mainMotor.step(STEPS\_PER\_TICK); |
|  | persistentData.currentPosition++; |
|  | } else { |
|  | mainMotor.step(-STEPS\_PER\_TICK); |
|  | persistentData.currentPosition--; |
|  | } |
|  | EEPROM.put(0, persistentData); |
|  |  |
|  | // If we've reached a floor, open the door and turn off the appropriate lights. This would be |
|  | // more efficient if we didn't loop, but rather, checked the "next expected" floor. |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | if (persistentData.currentPosition == persistentData.floorLevels[i]) { |
|  | floorDisplay.writeDigit(i + 1); |
|  | persistentData.currentFloor = i; |
|  | EEPROM.put(0, persistentData); |
|  | Serial.print("At floor "); |
|  | Serial.print(i + 1); |
|  | Serial.print(": "); |
|  | Serial.println(persistentData.currentPosition); |
|  |  |
|  | printStatus("AT FLOOR"); |
|  |  |
|  | // Check to see if someone wanted to go here; if so, stop |
|  | if (floorRequests[i] == EITHER || floorRequests[i] == BOTH || |
|  | (currentDirection == UP && floorRequests[i] == UP) || |
|  | (currentDirection == DOWN && floorRequests[i] == DOWN) || |
|  | i == destinationFloor) { |
|  | Serial.println("Stopping here!"); |
|  | floorTimer = FLOOR\_TIMER\_START; |
|  |  |
|  | // Figure out which lights to turn off |
|  | digitalWrite(floorLights[i], LOW); |
|  |  |
|  | if (currentDirection == UP) { |
|  | digitalWrite(upLights[i], LOW); |
|  | if (i == destinationFloor) { |
|  | digitalWrite(downLights[i - 1], LOW); |
|  | } |
|  | if (floorRequests[i] == BOTH) { |
|  | floorRequests[i] = DOWN; |
|  | } else { |
|  | floorRequests[i] = IDLING; |
|  | } |
|  | } else if (currentDirection == DOWN) { |
|  | digitalWrite(downLights[i - 1], LOW); |
|  | if (i == destinationFloor) { |
|  | digitalWrite(upLights[i], LOW); |
|  | } |
|  | if (floorRequests[i] == BOTH) { |
|  | floorRequests[i] = UP; |
|  | } else { |
|  | floorRequests[i] = IDLING; |
|  | } |
|  | } |
|  |  |
|  | calcDestination(); |
|  | printStatus("AFTER STOP"); |
|  | } |
|  | break; // Found a floor, no need to check the rest of them |
|  | } |
|  | } |
|  | } else if (currentDirection == IDLING) { |
|  | // No need to go anywhere |
|  | powerOffStepper(); |
|  | } |
|  | } |
|  |  |
|  | void calcDestination() { |
|  | destinationFloor = -1; |
|  |  |
|  | switch (currentDirection) { |
|  | case UP: |
|  | case IDLING: |
|  | for (int i = FLOORS - 1; i > 0; i--) { |
|  | if (floorRequests[i]) { |
|  | destinationFloor = i; |
|  | break; |
|  | } |
|  | } |
|  | break; |
|  |  |
|  | case DOWN: |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | if (floorRequests[i]) { |
|  | destinationFloor = i; |
|  | break; |
|  | } |
|  | } |
|  | break; |
|  | } |
|  |  |
|  | if (destinationFloor == -1 || destinationFloor == persistentData.currentFloor) { |
|  | // Nowhere to go |
|  | currentDirection = IDLING; |
|  | turnOffArrows(); |
|  | } else if (destinationFloor > persistentData.currentFloor) { |
|  | currentDirection = UP; |
|  | showUpArrow(); |
|  | } else if (destinationFloor < persistentData.currentFloor) { |
|  | currentDirection = DOWN; |
|  | showDownArrow(); |
|  | } |
|  | } |
|  |  |
|  | void printStatus(const char \*str) { |
|  | Serial.print("REQUEST STATUS: "); |
|  | Serial.println(str); |
|  | for (int i = FLOORS - 1; i >= 0; i--) { |
|  | Serial.print("Floor "); |
|  | Serial.print(i + 1); |
|  | Serial.print(": "); |
|  | Serial.println(floorRequests[i]); |
|  | } |
|  | Serial.print("Heading towards floor "); |
|  | Serial.println(destinationFloor + 1); |
|  | } |
|  |  |
|  | void powerOffStepper() { |
|  | digitalWrite(9, LOW); |
|  | digitalWrite(10, LOW); |
|  | digitalWrite(11, LOW); |
|  | digitalWrite(12, LOW); |
|  | } |
|  |  |
|  | void enterCalibrationMode() { |
|  | showCalibrationMode(); |
|  | digitalWrite(floorLights[0], HIGH); |
|  | digitalWrite(floorLights[FLOORS - 1], HIGH); |
|  | persistentData.currentPosition = 0; |
|  |  |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | floorDisplay.writeDigit(i + 1); |
|  | do { |
|  | if (digitalRead(floorButtons[0]) == HIGH) { |
|  | mainMotor.step(-STEPS\_PER\_TICK); |
|  | persistentData.currentPosition--; |
|  | } else if (digitalRead(floorButtons[FLOORS - 1]) == HIGH) { |
|  | mainMotor.step(STEPS\_PER\_TICK); |
|  | persistentData.currentPosition++; |
|  | } |
|  | } while (digitalRead(floorButtons[2]) != HIGH); |
|  | persistentData.floorLevels[i] = persistentData.currentPosition; |
|  | Serial.print("Floor "); |
|  | Serial.print(i + 1); |
|  | Serial.print(": "); |
|  | Serial.println(persistentData.currentPosition); |
|  |  |
|  | delay(500); // Allow button to debounce before moving on to next floor |
|  | } |
|  | digitalWrite(floorLights[0], LOW); |
|  | digitalWrite(floorLights[FLOORS - 1], LOW); |
|  | directionMatrix.clearDisplay(0); |
|  | persistentData.currentFloor = FLOORS - 1; |
|  | EEPROM.put(0, persistentData); |
|  |  |
|  | for (int i = 0; i < FLOORS; i++) { |
|  | floorRequests[i] = IDLING; |
|  | } |
|  | } |
|  |  |
|  | /\* |
|  | \* Matrix functions. Note that items are mirror-image left-to-right |
|  | \*/ |
|  | void showUpArrow() { |
|  | directionMatrix.setColumn(0, 0, B00011000); |
|  | directionMatrix.setColumn(0, 1, B00111100); |
|  | directionMatrix.setColumn(0, 2, B01011010); |
|  | directionMatrix.setColumn(0, 3, B10011001); |
|  | directionMatrix.setColumn(0, 4, B00011000); |
|  | directionMatrix.setColumn(0, 5, B00011000); |
|  | directionMatrix.setColumn(0, 6, B00011000); |
|  | directionMatrix.setColumn(0, 7, B00011000); |
|  | } |
|  |  |
|  | void showDownArrow() { |
|  | directionMatrix.setColumn(0, 0, B00011000); |
|  | directionMatrix.setColumn(0, 1, B00011000); |
|  | directionMatrix.setColumn(0, 2, B00011000); |
|  | directionMatrix.setColumn(0, 3, B00011000); |
|  | directionMatrix.setColumn(0, 4, B10011001); |
|  | directionMatrix.setColumn(0, 5, B01011010); |
|  | directionMatrix.setColumn(0, 6, B00111100); |
|  | directionMatrix.setColumn(0, 7, B00011000); |
|  | } |
|  |  |
|  | void showCalibrationMode() { |
|  | directionMatrix.setColumn(0, 0, B00000000); |
|  | directionMatrix.setColumn(0, 1, B00111100); |
|  | directionMatrix.setColumn(0, 2, B01000010); |
|  | directionMatrix.setColumn(0, 3, B00000010); |
|  | directionMatrix.setColumn(0, 4, B00000010); |
|  | directionMatrix.setColumn(0, 5, B01000010); |
|  | directionMatrix.setColumn(0, 6, B00111100); |
|  | directionMatrix.setColumn(0, 7, B00000000); |
|  | } |
|  |  |
|  | void setMatrixCorners(boolean value) { |
|  | directionMatrix.setLed(0, 0, 0, value); |
|  | directionMatrix.setLed(0, 0, 7, value); |
|  | directionMatrix.setLed(0, 7, 0, value); |
|  | directionMatrix.setLed(0, 7, 7, value); |
|  | } |
|  |  |
|  | void turnOffArrows() { |
|  | directionMatrix.clearDisplay(0); |
|  | } |