```cpp

/*-------------------------------------------------------------------------
Filename:     ArduinoSerial.cpp
Project:      IEEE SoutheastCon Hardware Competition 2019
School:       Auburn University
Organization: Student Projects and Research Committee (SPARC)
Description:  Communicates with an Arduino from the Raspberry Pi 3 B+ over USB.
Controls 2 drive motors and 3 steppers.  Speed ranges are from -127 to 127.
-------------------------------------------------------------------------*/
#include "ArduinoSerial.h"
#include <stdexcept>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <cerrno>
#include <cstring>    //for strerror()
#include <sstream>
#include <iostream>
#include <string>

// Constants
#define DEBUG_TEXT 0
const char serialPort::typicalPortName[] = "/dev/ttyUSB1";

// Namespaces
using namespace std;

// Variables
int leftDriveSpeed = 0;
int rightDriveSpeed = 0;
int gatePos = 0;
int flagPos = 0;
string LCDtext = "Connected!";
string buttonState = "0";
int clearButtonState = 0;
string mode = "-1";


serialPort::serialPort(const char* portName) {
  fileHandle = open(portName, O_RDWR | O_NOCTTY | O_NDELAY);
  if(fileHandle == -1) {
    throw std::runtime_error(string("Error opening port: ") + strerror(errno));
  }
  if(!isatty(fileHandle)) {
    close(fileHandle);
    throw std::runtime_error("Port is not a serial device.");
  }

  cfmakeraw(&config);     //Sets various parameters for non-canonical mode; disables parity

  cfsetospeed (&config, B38400);    //Baud rate
  cfsetispeed (&config, B38400);

  config.c_cflag     &=  ~CSTOPB;    //One stop bit

  config.c_cflag     |=  CREAD | CLOCAL;
  config.c_cflag     &=  ~CRTSCTS;           // no flow control

  config.c_cc[VMIN]   =  0;
```

```
60    config.c_cc[VTIME]  =  0;
61
62    if(tcsetattr(fileHandle, TCSANOW, &config) != 0) {
63      close(fileHandle);
64      throw std::runtime_error("Setting attributes failed.");
65    }
66    usleep(1000*1000*1); // wait 1 sec for arduino to reset
67    tcflush(fileHandle, TCIFLUSH);    //clear input buffer
68    usleep(1000*1000*1); // wait 1 sec for arduino to reset
69  }
70
71  void serialPort::write(string text) {
72    ssize_t bytes_written = ::write(fileHandle, text.c_str(), text.length());
73  }
74
75  void serialPort::write(char data[], int length) {
76    ssize_t bytes_written =  ::write(fileHandle, data, length);
77  }
78
79  string serialPort::read() {
80    string input;
81    int bytes = available();
82    input.resize(bytes);
83    ssize_t bytes_written = ::read(fileHandle, const_cast<char*>(input.data()) ,
  bytes);    //The const cast is less than ideal
84    return(input);
85  }
86
87  int serialPort::available() {
88    int bytes;
89    ioctl(fileHandle, FIONREAD, &bytes);
90    return(bytes);
91  }
92
93  // Updates the motor speeds according to the state variables
94  string serialPort::updateArduino() {
95    signed char char1 = (signed char)(leftDriveSpeed);
96    signed char char2 = (signed char)(rightDriveSpeed);
97    unsigned char char3 = (unsigned char)(gatePos);
98    unsigned char char4 = (unsigned char)(flagPos);
99    unsigned char char6 = (unsigned char)(clearButtonState);
100   string newText = LCDtext;
101   newText.resize(32 , ' ');
102   unsigned char char5 = char1 + char2 + char3 + char4 + char6 + 1;
103   char startChar[1] = {'<'};
104   signed char motorData[2] = {char1,char2};
105   unsigned char servoData[4] = {char3, char4, char6, char5};
106   char endChar[1] = {'>'};
107   if (DEBUG_TEXT){
108     cout << "Sending to Arduino: " << startChar[0] << (int)char1 << "," <<
  (int)char2 << "," << (int)char3  << "," << (int)char4 << "," << (int)char6 << ",";
109     cout << (int)char5 << ","  << newText << "," << endChar[0] <<  endl;
110    // cout << "Total bytes: " << (6+newText.length()) << endl;
111   }
112   ssize_t bytes_written = ::write(fileHandle, startChar, 1);
113   bytes_written = ::write(fileHandle, motorData, 2); //Send messages
114   bytes_written = ::write(fileHandle, servoData, 4); //Appended checksum
115   bytes_written = ::write(fileHandle, newText.c_str(), newText.length());
116   bytes_written = ::write(fileHandle, endChar, 1);
117  // if (DEBUG_TEXT){ //Output text characters
```

```
118  //   char qqqq[32];
119  //   memcpy(qqqq,newText.c_str(),32);
120  //   for(int i = 0; i<32; i++)cout << (int)qqqq[i] << " ";
121  //   cout << endl;
122  // }
123    string received = read();
124    string delim = ","; //Pick out 6th value to find buttonState
125    auto start = 0U;
126    auto end = received.find(delim);
127    int value = 0;
128    if (DEBUG_TEXT){
129    cout << "Recieved from Arduino: ";
130    }
131    while (end != string::npos){
132    value++;
133    if(value==8){
134      buttonState = received.substr(start, end - start);
135      if (DEBUG_TEXT){
136        cout << "ButtonState: " << buttonState << ",";
137      }
138    }
139    else if(value==9){
140      mode = received.substr(start, end - start);
141      if (DEBUG_TEXT){
142        cout << "Mode: " << mode;
143      }
144      break;
145    }
146    else if (DEBUG_TEXT){
147       cout << received.substr(start, end - start) << ",";
148    }
149    start = end + delim.length();
150    end = received.find(delim, start);
151    }
152    if(DEBUG_TEXT)
153      cout << endl;
154    return buttonState;
155  }
156
157  void serialPort::turnLeft(int speed){
158    if(speed < 0 || speed > 127)
159      throw out_of_range("Motor speed must be between 0 and 127.");
160    leftDriveSpeed = speed;
161    rightDriveSpeed = -speed;
162  }
163
164  void serialPort::turnRight(int speed){
165    if(speed < 0 || speed > 127)
166      throw out_of_range("Motor speed must be between 0 and 127.");
167    leftDriveSpeed = -speed;
168    rightDriveSpeed = speed;
169  }
170
171  void serialPort::goForward(int speed){
172    if(speed < 0 || speed > 127)
173      throw out_of_range("Motor speed must be between 0 and 127.");
174    leftDriveSpeed = -speed;
175    rightDriveSpeed = -speed;
176  }
177
```

```
178 void serialPort::goBackward(int speed){
179   if(speed < 0 || speed > 127)
180       throw out_of_range("Motor speed must be between 0 and 127.");
181   leftDriveSpeed = speed;
182   rightDriveSpeed = speed;
183 }
184 void serialPort::drive(int left, int right){
185   leftDriveSpeed = left;
186   rightDriveSpeed = right;
187 }
188 void serialPort::stopMotors(){
189   leftDriveSpeed = 0;
190   rightDriveSpeed = 0;
191 }
192
193 void serialPort::moveGate(int pos){
194   if(pos < 0 || pos > 180)
195     throw out_of_range("Servo position must be between 0 and 180.");
196   gatePos = pos;
197 }
198
199 void serialPort::moveFlag(int pos){
200   if(pos < 0 || pos > 180)
201     throw out_of_range("Servo position must be between 0 and 180.");
202   flagPos = pos;
203 }
204
205 void serialPort::updateLCD(string text){
206   LCDtext = text;
207   if(DEBUG_TEXT){
208    cout << "New LCD text: " << text << endl;
209   }
210 }
211
212 int serialPort::getMode(){
213   if(mode == "0")return 0;
214   if(mode == "1")return 1;
215   if(mode == "2")return 2;
216   if(mode == "3")return 3;
217   return 4;
218 }
219
220 int serialPort::getButtonState(){
221   int currentState = 0;
222   if (buttonState == "1"){
223     clearButtonState = 1;
224     buttonState = "0";
225     currentState = 1;
226   }
227   else{
228     clearButtonState = 0;
229     currentState = 0;
230   }
231   return currentState;
232 }
233
234
235 serialPort::~serialPort() {
236   cout << "Disconnecting from Arduino..." << endl;
237   updateLCD("Disconnected..");
```

```
238   updateArduino();
239   close(fileHandle);
240 }
241
```