## Disclaimer:

Low-cost Open-source Robotics Educational (LORE) kits are provided "as is" with no warranties whatsoever, including any warranty of merchantability, non - infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification or sample.

LORE kits (and manufacturers and distributors) disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in any documents and files and software and no license, express or implied, by LORE kits or otherwise, to any intellectual property rights is granted herein. LORE kits (and manufacturers and distributors) assumes no responsibility or liability for any errors or inaccuracies that may appear in any documentation or files or any software that may be provided.

The information in any documents or files is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by LORE kits (and manufacturers and distributors).

# Contents



# Table of Figures



# Editions

Edition 1.

Edition 1.01. Added introductory programming examples, i.e. Getting Started. Added additional information regarding new functions, e.g. turn() and setTurnTime().

# 1. Low-Cost Open-Source Robotics Education (LORE) kit Programming Guide:

Programming the LORE robot can be broken into three levels of difficulty:

- **Beginner Programming Level**. Suitable for students with no programming experience. No computer experience required.
- **Intermediate Programming Level**. Suitable for students with some programming knowledge. Students should be familiar with computers and how to use an Internet browser, keyboard and mouse.
- **Advanced Programming Level**. Suitable for students with programming experience in C/C++. Students should be computer literate and familiar with Windows 7/8.

To familiarise students with programming, the following section provides a primer on programming.

# 2. Programming Primer:

There are a number of excellent resources for learning to program, e.g. LearnCpp.com. Here, we provide a very basic, and limited, introduction to some key concepts: variables and functions, which are used in the Intermediate and Advanced Programming Levels.

For a complete introduction to variables and functions, see the tutorials available from LearnCpp:

http://www.learncpp.com.

**Variables:**

"A variable in C/C++ is a name for a piece of memory that can be used to store information" (C++, 2014). A variable can be thought of as a box, where information can be put.

Here's an example of declaring an integer variable:

```
int x = 0;
```

The variable's name is x and is set to zero. Keyword int sets the variable type to integer. The equal sign is used to specify what integer x is equal to. The number zero sets x's value.

For a complete list of variable types, see the following link:

http://en.wikipedia.org/wiki/C_data_types

**Functions:**

"A function is a sequence of statements designed to do a particular job" (C++, 2014). In C/C++, every program requires a main() function; however, for Arduino boards, there must be a setup() function and a loop() function.

Here's an example of declaring a function:

```
void print(char* string) {
	Serial.print(string);
}
```

The function's name is print. Function print() takes a single parameter: string, which is an array of characters. Function print() calls the print() member function, or method, of an instance of a Serial Port class, Serial. Function print() does not return any value; however, functions can return any standard data type.

# 3. Beginner Programming Level Guide:

The Beginner Programming Level is suitable for students that have no prior knowledge of programming. Programming the LORE robot is done by pressing the PROGRAM button followed by pressing a series of LEFT, RIGHT, FRONT, and BACKWARD directional buttons. Once a program is finished, pressing the SELECT button saves the program. Each time the directional buttons are pressed, the LCD panel will display the button pressed. By default the maximum program size is 50 steps; however, this can be changed. When the program is full, the LCD will display "PROGRAM FULL". Pressing the SELECT button a second time executes the program. The program will run through each pre-programmed step until it is finished. By default, each step will execute for one second (FORWARD and BACKWARD movements) or 90° (LEFT or RIGHT turns). Pressing the SELECT button will re-run the program. Pressing the PROGRAM button again will re-enter programming mode. When re-entering programming mode, the previous program is overwritten.

The following figure (Figure 1) outlines the Beginner Programming Level structure:

**Press the PROGRAM button.**
Press the PROGRAM button to enter programming mode.
↓
**Enter the PROGRAM.**
Press the LEFT, RIGHT, FORWARD, or BACKWARD buttons.
Repeat this step till desired movements are loaded.
↓
**Press the SELECT button.**
Press the SELECT button to save the program.
↓
**Press the SELECT button.**
Press the SELECT button to run the program.
↓
Wait for program to run.

Figure 1: Beginning Programming Level procedure.

## 3.1. Beginner Programming Level Example 1:

Consider a student, Marty, who wants to make the LORE robot move forward for 10 seconds, turn left, move forward for two seconds, and then stop.

To program the LORE robot to do this, the student should do the following:

1. **Press the PROGRAM button.** This will enter programming mode.
2. **Press the FORWARD button.** This will make the LORE robot move forward for one second.
3. **Repeat Step 2 9 more times.** Pressing the FORWARD button an additional nine more times will program the robot to move forward for an additional nine seconds.
4. **Press the LEFT button.** This will make the robot turn 90° to the left.
5. **Press the FORWARD button.** This will make the LORE robot move forward for one second.
6. **Repeat Step 5 one more time.** Pressing the FORWARD button one more time will program the robot to move forward for an additional one second.
7. **Press the SELECT Button.**

To run the program, the student should press the SELECT button again.

# 4. Intermediate Programming Level Guide:

The Intermediate Programming Level is suitable for students who have some prior knowledge of programming and are familiar with computers, i.e. able to use a mouse and keyboard for typing and data entry in a Windows 7/8 environment, as well as, browsing the Internet using an Internet browser. Students are provided with a simple set of functions, which they can call to program the robot to move (FORWARD and BACKWARD) and turn 90° (LEFT and RIGHT). Students can also access the LORE robot's FRONT SENSOR, LEFT SENSOR, and RIGHT SENSOR ultrasonic sensors to measure distances from the robot.

## 4.1. Getting Started:

Before being able to program the robot, students will need to download the Arduino integrated development environment (IDE) and the ROBOT library file. The IDE can be downloaded for free from the following site:

http://arduino.cc/

Browse to the Download section and download the latest, stable version: as of May, 2014 it is version 1.0.5. The Windows Installer will automatically install all the software and drivers necessary to program the LORE robot's microcontroller: the Arduino Uno. By default the IDE will install to C:\Progam Files\Arduino; however, you may install this to any directory. When installing the Arduino IDE you will need administrator rights to install software and driver. For supplementary instructions on installing the Arduino IDE, refer to the Arduino website.

Once the Arduino IDE is installed, you can make use of a library created for the LORE robot and the Arduino board. Download ROBOT.zip from the following link:

https://sourceforge.net/projects/lorerobotics/files/.

Extract ROBOT.zip to the Arduino user library folder. To extract the file, right click the file and left click Extract All…. By default the Arduino user library folder will be in the default Arduino workspace within the documents directory, e.g. C:\Users\...\Documents\Arduino\libraries (NB. "…" corresponds to the current user name; this will differ from machine to machine).

To use the library, open the Arduino IDE and left click Select>Add Library…>ROBOT. If ROBOT.zip has been extracted to the correct file, the library ROBOT will be listed under the Contributed libraries.

## 4.2. Programming the Arduino Board:

The Arduino board is a simple-to-use and easy-to-program development board. The LORE robot uses the Uno model (see Figure 2).


Figure 2: Arduino Uno R3 Board

For tutorials on programming Arduinos, have a look at the following link:

http://arduino.cc/en/Tutorial/HomePage

At the least, the following code is necessary for programming the Arduino board:

```
void setup()
{
	// put your setup code here, to run once:

}

void loop()
{
	// put your main code here, to run repeatedly:

}
```

The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board. After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board. Program logic should be placed in loop(), which will continuously execute the program.

**Intermediate Programming Level Tip 1:**

*When uploading the programme to the LORE robot's Arduino, you'll need to identify which COM port to use. Typically COM 1/2 refers to the hardware serial ports on older machines, so you'll find the Arduino's COM port will be COM 3 or higher. You can be sure which COM port it is by using Window's Device Manager. This can be found by pressing the Windows Start key and typing "Device Manager", then selecting Device Manager under the found items. Browsing to the Ports (COM & LPT) menu and expanding it, you can see what the Arduino's COM port is.*

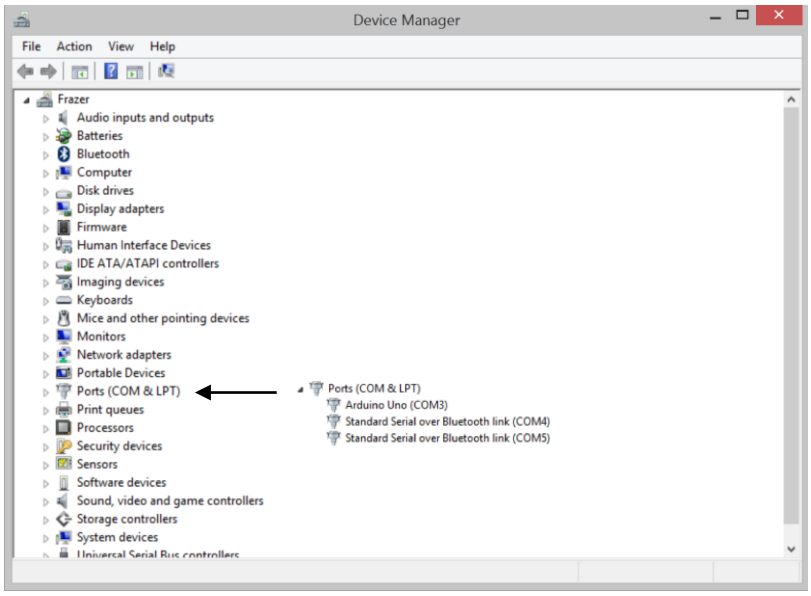


Figure 3: Windows Device Manager. (Insert) Drop down menu showing which port the Arduino is connected to.

## 4.3. Using the ROBOT Library:

The ROBOT library provides a number of easy-to-use functions that allow for the LORE robot to be controlled. The ROBOT library functions are as follows:

- **`bool initialise().`** The initialise() function sets all the Arduino board's pins direction (INPUT or OUTPUT) and sets their states (HIGH or LOW). When the robot is initialised, the function returns TRUE.
- **`bool buttonPressed(int button).`** The buttonPressed() function is used to see if one of the LORE robot's buttons have been pressed. If the button is pressed, the function buttonPressed returns TRUE; else, it returns FALSE.
- **`bool clear().`** The clear() function clears the LORE robot's program. When the robot's program is cleared, the function returns TRUE.
- **`bool move(int direction).`** The move() function moves the LORE robot in a specific direction (FORWARD or BACKWARD): parameter direction specifies the movement it makes. When move finished, the function returns TRUE. If direction is not FORWARD or BACKWARD, move() returns FALSE.
- **`bool turn(int direction)`**. The turn() function turns the LORE robot in a specific direction (LEFT and RIGHT): the parameter direction defines the turn it makes. When turn() finished, the function returns TRUE. If direction is not LEFT or RIGHT, turn() returns FALSE.
- **`bool setTurnTime(int direction, int time).`** The setTurnTime() sets the time the robot turns in a specific direction (LEFT or RIGHT) for a specific time: the parameter direction defines which turn and parameter time specifies for how long. When setTurnTime() finished, the function returns TRUE. If direction is not LEFT or RIGHT, setTurnTime() returns FALSE.
- **`bool program(int direction).`** The program() function allows the LORE robot to be programmed in a similar way to the beginner programming use-case. If the program size is less than 50, the function returns TRUE for successfully programming direction = LEFT | RIGHT | FORWARD | BACKWARD; else, it returns FALSE.
- **`bool run().`** The run() function runs the program set using the program function. If program size is greater than zero, the function returns TRUE; else, it returns FALSE.
- **`bool writeLCD(char* string).`** The writeLCD() function prints a series of characters to the LORE robot's LCD panel. The maximum number of characters that can be shown is 16. If the string is not empty, the function returns TRUE on completion; else, it returns FALSE.
- **`long measureDistance(int sensor).`** The measureDistance() function samples the LORE robot's ultrasonic sensors. If the robot successfully makes a measurement, measureDistance() returns the distance in centimetres; else, it returns -1;

All these functions are provided via a class. To access these functions, create an instance of the ROBOT class. For example:

```
ROBOT marty;
```

Here, marty is the name of the instance of the ROBOT class. Think of the ROBOT describing the LORE robot and the instance, i.e. marty, the name of the specific robot. The instance could be called anything sensible, e.g. Judy, mark, WALLE, etc. Only one instance needs to be used to control the LORE robot.

To gain access to the initialise(), buttonPressed(), clear(), move(), program(), run(), and writeLCD() functions, you need to add '.' to the instance name and type the function name. For example:

```
marty.initialise();
marty.buttonPressed(PROGRAM);
marty.clear();
```

```
marty.move(FORWARD);
marty.program(FORWARD);
marty.run();
marty.writeLCD("HELLO");
```

Note, for buttonPressed() and and program(), the acceptable inputs are: PROGRAM, SELECT, LEFT, RIGHT, FORWARD, and BACKWARD. Only these keywords will work.

For move(), the acceptable inputs are: LEFT, RIGHT, FORWARD, and BACKWARD. Only these keywords will work.

For writeLCD(), you can print any string of characters (as long as they're English and the string length is less than 16).

The following figure (Figure 4) outlines the intermediate programming structure:

**Open the Arduino IDE.**
Open the Arduino IDE.
↓
**Add the ROBOT library.**
Left click Select>Add Library…>ROBOT.
This will add the Servo, LiquidCrystal, and ROBOT header files automatically.
↓
**Enter the program.**
First, create an instance of the ROBOT class; next, Using initialise(), buttonPressed(), clear(), move(), program(), run(), and writeLCD(), program the robot to carry out a desired set of movements, read data from sensors, and wait for input from the user.
↓
**Upload the program to the LORE robot.**
Turn on the LORE robot and plug in the USB cable. Press Upload in the Arduino IDE.
↓
Wait for program to run.

Figure 4: Intermediate Programming Level procedure.

### 4.4. Intermediate Programming Level Example 1:

Consider a student, Marty, who wants to make the LORE robot move forward for 10 seconds, turn left, move forward for two seconds, and then stop.

To program the LORE robot to do this, the student should do the following:

1. **Open the Arduino IDE.**
2. **Add the ROBOT library.**
3. **Enter the following program:**

```
#include <ROBOT.h>
#include <Servo.h>
#include <LiquidCrystal.h>

ROBOT marty;

void setup()
{

   marty.initialise();
```

```
}

void loop()
{

    if (marty.buttonPressed(PROGRAM)) {

            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);
            marty.move(FORWARD);

            marty.move(LEFT);

            marty.move(FORWARD);
            marty.move(FORWARD);

    }


}
```

Here, we created an instance of the ROBOT class named marty. Then we initialised marty using the initialise() function. We then checked to see if the PROGRAM button was pressed; if it was, we made the LORE robot move; else, we waited till the PROGRAM button is pressed.

4. **Upload the program to the LORE robot.**

Once the program has been uploaded to the LORE robot's Arduino board, the program will begin to execute immediately.

**Intermediate Programming Level Tip 2:**

*When writing a program for the LORE robot, don't forget to use the robot's buttons. You can make the robot wait for you to press a button, if you don't want it to run your program straight away.*

**Intermediate Programming Level Example 2:**

Consider the same student, Marty, who now wants to make the LORE robot move forward for 10 seconds, but stop if an object is 20 cm in front of the robot.

To program the LORE robot to do this, Marty should do the following:

1. **Open the Arduino IDE.**
2. **Add the ROBOT library.**
3. **Enter the following program:**

```
#include <ROBOT.h>
#include <Servo.h>
#include <LiquidCrystal.h>

ROBOT marty;

void setup()
```

```
{

    marty.initialise();

}

void loop()
{

    if (robot.buttonPressed(PROGRAM)) {

           for (int i = 0; i < 10; i++) {

                  marty.move(FORWARD);

                  long distance = 0;
                  distance = marty.measureDistance(FRONT_SENSOR);

                  if (distance <= 20) {
                         break;
                  }

           }

    }


}
```

Here, we created an instance of the ROBOT class named marty. Then we initialised marty using the initialise() function. We then checked to see if the PROGRAM button was pressed; if it was, we made the LORE robot move FORWARD for 10 seconds; else, we waited till the PROGRAM button is pressed. If the FRONT_SENSOR measured a distance less than or equal to 20 cm, the robot stopped moving.

4. **Upload the program to the LORE robot.**

Once the program has been uploaded to the LORE robot's Arduino board, the program will begin to execute immediately.

# 5. Advanced Programming Level Guide

The Advanced Programming Level is suitable for students with programming experience in C/C++. Students should be computer literate and familiar with Windows 7/8. The Advanced Programming Level builds upon the Intermediate Programming Level. Students should download the Arduino IDE, as discussed in the Intermediate Programming Level instructions; however, it is not necessary for them to use the ROBOT library as the template provided provides the same functionality, but uses standard Arduino and C-based functions.

At this level, students should use the Arduino board's standard functions, e.g. pinMode(), digitalRead(), digitalWrite(); as well as, control structures, e.g. if, if…else, for, while, and break, as outlined on Arduino's reference site:

http://arduino.cc/en/Reference/HomePage.

The following functions will allow for direct control of the Arduino board's pins:

- `void pinMode(int pin)`. The pinMode() function defines whether an Arduino board's pin is either an INPUT or an OUTPUT pin. Data is read from an INPUT; data is written to an OUTPUT.
- `void digitalWrite(int pin, int value)`. The digitalWrite() function sets the state of an Arduino board's pin to either HIGH or LOW, which equates to 1 or 0, respectively.
- `bool digitalRead(int pin)`. The digitalRead() function reads the state of an INPUT pin, returning HIGH if the voltage on the pin is greater than 2.5 V; else, it returns LOW.

**Advanced Programming Level Tip 1:**

*The LORE robot makes use of internal 10 kΩ pull-up resistors, which means the state of the digital pins will normally be HIGH. To read if a button is pressed, check if the pin's state is LOW, e.g. if(digitalRead(FORWARD) == LOW){ }.*

**Advanced Programming Level Tip 2:**

*To enable the LORE robot's internal 10 kΩ pull-up, set the Arduino board's pins to INPUTs and set them HIGH, e.g. pinMode(FORWARD, INPUT); digitalWrite(FORWARD, HIGH);*

The following figure (Figure 5) outlines the intermediate programming structure:

**Open the Arduino IDE.**
Open the Arduino IDE.
↓
**Enter the program.**
Use the Advanced Programming Level template.
Using the functions move(), readButton(), and readSensor(), program the LORE robot to execute a desired set of movements. Initialise
↓
**Upload the program to the LORE robot.**
Turn on the LORE robot and plug in the USB cable. Press Upload in the Arduino IDE.
↓
Wait for program to run.

Figure 5: Advanced Programming Level procedure.

## 5.1. Advanced Programming Level Template:

Consider a student, Marty, who now wants to program the robot (at a more advanced level) himself; rather than use the ROBOT library.

To program the LORE robot, Marty should use the following template:

```
/*
Headers. Header files are declared here. The #include keyword is used to include a *.h
header file.
*/

#include <Servo.h>
#include <LiquidCrystal.h>

/*
Defines. Defines are declared here. Defines aren't variables; rather they're
statements which replace key words with values when the program is compiled.
*/

#define BACKWARD 7
#define FORWARD 8
#define RIGHT 9
#define LEFT 10
#define SELECT 11
#define PROGRAM 12

#define RS A0
#define E A1
#define D4 A2
#define D5 A3
#define D6 A4
#define D7 A5

#define FRONT_SENSOR 2
#define LEFT_SENSOR 3
#define RIGHT_SENSOR 4

#define LEFT_MOTORS 5
#define RIGHT_MOTORS 6

/*
Global Variables. Global variables are declared here. Format: variableType
variableName; or variableType variableName = defaultValue;
*/

long debounceDelay = 50;

long programDebounceTime = 0;
long selectDebounceTime = 0;
long leftDebounceTime = 0;
long rightDebounceTime = 0;
long forwardDebounceTime = 0;
long backwardDebounceTime = 0;

int programState = HIGH;
int programPrevState = HIGH;
int selectState = HIGH;
int selectPrevState = HIGH;
int leftState = HIGH;
int leftPrevState = HIGH;
```

```
int rightState = HIGH;
int rightPrevState = HIGH;
int forwardState = HIGH;
int forwardPrevState = HIGH;
int backwardState = HIGH;
int backwardPrevState = HIGH;

long frontSensorValue = 0;
long leftSensorValue = 0;
long rightSensorValue = 0;

char programString[50] = {};
int programSet = 0;
int programSize = 0;

/*
Class instances. Class objects are declared here.
*/

LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
Servo leftMotors, rightMotors;

/*
Function Declarations. Function prototypes are declared here. Format: returnType
functionName(parameterType parameterName, ...);
The function contents are defined after void loop().
*/

bool move(int direction);
bool readButton(int button, int &buttonState, int &buttonPrevState, long
&debounceTime);
bool readSensor(int sensor, long &distance);

/*
Function:     void setup().
Returns:      None.
Parameters:   None.
Description:  Setup initialises the program's variables. Here the pinMode's of the
PROGRAM through to BACKWARD buttons are set as inputs and then internal pull up
resistors enables by writing a HIGH logic level to the buttons' associated pins.
Sensors are configured as outputs and their states set to the LOW logic level. Servo
motors are attached to LEFT_MOTORS and RIGHT_MOTORS pins and set to a neutral
position. Lastly, the serial port is opened using a default baud rate of 9600 and 8N1
parameters (8 data-bits, no parity, 1 stop bit).
*/

void setup()
{

       // Initialise the LCD screen. Set size at 16 channels x 2 rows.

       lcd.begin(16, 2);
       lcd.clear();
       lcd.home();

       // Initialise the pushbuttons. Enable internal pull-ups. Logic LOW = button
       pressed; else HIGH.

       pinMode(PROGRAM, INPUT);
       digitalWrite(PROGRAM, HIGH);

       pinMode(SELECT, INPUT);
```

```
        digitalWrite(SELECT, HIGH);

        pinMode(LEFT, INPUT);
        digitalWrite(LEFT, HIGH);

        pinMode(RIGHT, INPUT);
        digitalWrite(RIGHT, HIGH);

        pinMode(FORWARD, INPUT);
        digitalWrite(FORWARD, HIGH);

        pinMode(BACKWARD, INPUT);
        digitalWrite(BACKWARD, HIGH);

        // Initialise the sensors. Configure as outputs to start with.

        pinMode(FRONT_SENSOR, OUTPUT);
        digitalWrite(FRONT_SENSOR, LOW);

        pinMode(LEFT_SENSOR, OUTPUT);
        digitalWrite(LEFT_SENSOR, LOW);

        pinMode(RIGHT_SENSOR, OUTPUT);
        digitalWrite(RIGHT_SENSOR, LOW);

        // Initialise servo motors.

        leftMotors.attach(LEFT_MOTORS);
        leftMotors.write(127);

        rightMotors.attach(RIGHT_MOTORS);
        rightMotors.write(127);
}

/*
Function:     void loop().
Returns:      None.
Parameters:   None.
Description:  Loops continuously, executing program logic.
*/

void loop()
{


}

/*
Function:     bool drive(int direction).
Returns:      bool.
Parameters:   int direction.
Description:  Reads in the direction the robot needs to move in. If direction is
FORWARD, read front sensor and drive for 1000 ms. If sensorValue is less than a
threshold, stop and move onto next instruction. If direction is !FORWARD, move for
1000 ms.
*/

bool move(int direction) {

        if (direction == FORWARD) {

                leftMotors.write(0);
```

```
        rightMotors.write(255);

        unsigned long time = 0;
        time = millis();

        while ((millis() - time) < 1000) {

                readSensor(FRONT_SENSOR, frontSensorValue);

                if (frontSensorValue < 5) {
                        break;
                }

        }

        leftMotors.write(127);
        rightMotors.write(127);

        return true;

}
else if (direction == BACKWARD) {

        leftMotors.write(255);
        rightMotors.write(0);

        delay(1000);

        leftMotors.write(127);
        rightMotors.write(127);

        return true;

}
else if (direction == LEFT) {

        leftMotors.write(255);
        rightMotors.write(255);

        delay(900);

        leftMotors.write(127);
        rightMotors.write(127);

        return true;

}
else if (direction == RIGHT) {

        leftMotors.write(0);
        rightMotors.write(0);

        delay(900);

        leftMotors.write(127);
        rightMotors.write(127);

        return true;

}

return false;
```

```
}


/*
Function:     bool readButton(int button, int &buttonState, int &buttonPrevState, long
&debounceTime).
Returns:      bool.
Parameters:   int button, int &buttonState, int &buttonPrevState, long &debounceTime.
Description:
*/

bool readButton(int button, int &buttonState, int &buttonPrevState, long
&debounceTime) {

	int reading = digitalRead(button);

	if (reading != buttonPrevState) {
		debounceTime = millis();
	}

	if ((millis() - debounceTime) > debounceDelay) {
		if (reading != buttonState) {
			buttonState = reading;

			if (buttonState == LOW) {
				return true;
			}
		}
	}

	buttonPrevState = reading;

	return false;

}

/*
Function:     bool readSensor(int sensor, long &distance).
Returns:      bool.
Parameters:   int sensor, long &distance.
Description:
*/

bool readSensor(int sensor, long &distance) {

	pinMode(sensor, OUTPUT);
	digitalWrite(sensor, LOW);
	delay(10);
	digitalWrite(sensor, HIGH);
	delay(10);
	pinMode(sensor, INPUT);
	int value = 0;
	value = pulseIn(sensor, HIGH);
	distance = value / 29 / 2;

	if (distance != 0) {
		return true;
	}

	return false;
}
```

In this template, the defines, e.g. `#define BACKWARD 7`, map keywords to a pin number. These numbers correspond to the pins on the LORE robot's Arduino board and the PROGRAM, SELECT, LEFT, RIGHT, FORWARD, and BACKWARD buttons; the LCD panel's data pins; and the FRONT_SENSOR, LEFT_SENSOR, and RIGHT_SENSOR data pins. The advantage of using #define is that it makes it easier to know what pin is associate to what button, etc.

The global variables, e.g. debounceDelay, programDebounceTime, programState, and programPrevState, are used to de-bounce the PROGRAM, SELECT, LEFT, RIGHT, FORWARD, and BACKWARD buttons. The variables frontSensorValue, leftSensorValue, and rightSensorValue are used to store the data measured from the LORE robot's FRONT_SENSOR, LEFT_SENSOR, and RIGHT_SENSOR.

# Bibliography

C++, L. (2014, May 30). *Learn C++*. Retrieved from Learn C++ A First Look at Variables: http://www.learncpp.com/cpp-tutorial/13-a-first-look-at-variables/