ARMbot Suite Integration

Brad Riske and Lucas Jacobs
Math and Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57718
bradriske@gmail.com, lucas.jacobs@mines.sdsmt.edu

Abstract

Simple robotics kits, like the Lego Mindstorms NXT, have been widely successful as instructional tools for elementary students - almost anyone of any age can use them effectively, the results of a simple program are readily understood by watching a robot in action, and they're fun to use. However, most click-and-drag toolkits are limited by their simplicity, and many other robots require familiarity with embedded systems and languages like C.

The ARMbot was designed as a transitional teaching aid. They use ARM7 processors and include an API that freshman CS students can easily access. The ARMbot suite assists students further by providing an emulation environment and a point-and-click design tool that generates C code. This term's project will further enhance the ARMbot suite's functionality by allowing this generated code to be edited, with changes reflected in the design tool. Both projects will also be integrated into a single IDE (Code::Blocks).

1 Introduction

The ARMbot Suite Integration project provides freshmen CS students, elementary students interested in robotics programming, and hobbyists with a comprehensive set of tools that share a single development environment. With these utilities, users can develop, test and simulate robotics code with little or no formal programming experience required.

This SDSM&T 2009 senior design project reimplements two projects from 2008 as plugins for an open-source, cross-platform development environment (Code::Blocks) while extending functionality.

2 ARMbot

Dr. Jeff McGough (an instructor at SDSM&T) obtained a NASA grant to increase interest in computer science and engineering. Using this grant, Dr. McGough built several programmable robots with off-the-shelf parts for educational and promotional purposes. Each ARMbot, named for the ARM7 embedded processor that they use, costs around $120 and can perform a variety of simple robotic functions. They can be programmed to navigate around a room, while following black lines on the floor using an optical sensor, or using ultrasonic detectors to detect (and avoid) obstacles.

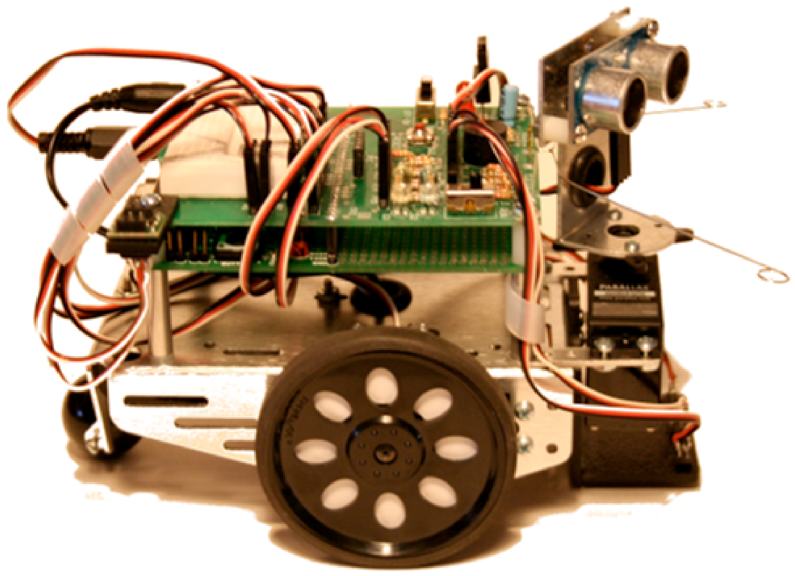
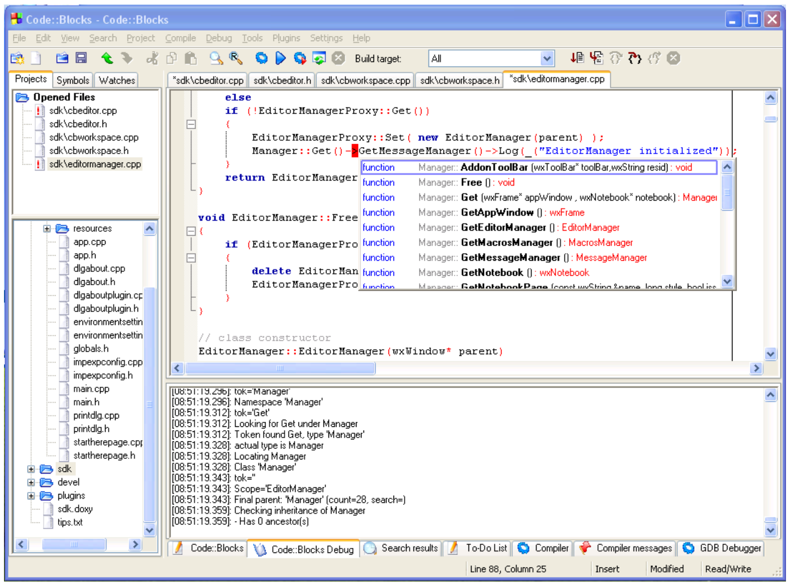


Figure 1: Picture of an ARMbot.

These robots have been used as development platforms for Dr. McGough's introductory CSC150 class, which is a required course for most majors at SDSM&T. Although controlling a robot automatically has proven to be more effective in helping students to directly interact with the course material, the lack of a single, easy-to-use IDE and the need to share a limited number of ARMbots for debugging purposes made the coursework less accessible for the freshman students.

3 Code::Blocks IDE

The goal for this project is to integrate several ARMbot-related utilities into one convenient package which aspiring programmers can use to develop, test and simulate code for use with ARMbots. Code::Blocks was chosen for its open-source licensing, flexible plugin API and portability. Like both preexisting projects, Code::Blocks uses wxWidgets, which facilitated integration and provides a consistent look-and-feel between the IDE and the individual plugins.



```
    else
    if (!EditorManagerProxy::Get())
    {
        EditorManagerProxy::Set( new EditorManager(parent) );
        Manager::Get()->GetMessageManager()->Log(_("EditorManager initialized"));
    }
    return EditorManager
}

void EditorManager::Free
{
    if (EditorManagerPro
    {
        delete EditorMan
        EditorManagerPro
    }
}

// class constructor
EditorManager::EditorManager(wxWindow* parent)
```

```
function    Manager:: AddonToolBar (wxToolBar* toolBar,wxString resid) : void
function    Manager:: Free () : void
function    Manager:: Get (wxFrame* appWindow , wxNotebook* notebook) : Manager
function    Manager:: GetAppWindow () : wxFrame
function    Manager:: GetEditorManager () : EditorManager
function    Manager:: GetMacrosManager () : MacrosManager
function    Manager:: GetMessageManager () : MessageManager
function    Manager:: GetNotebook () : wxNotebook
```

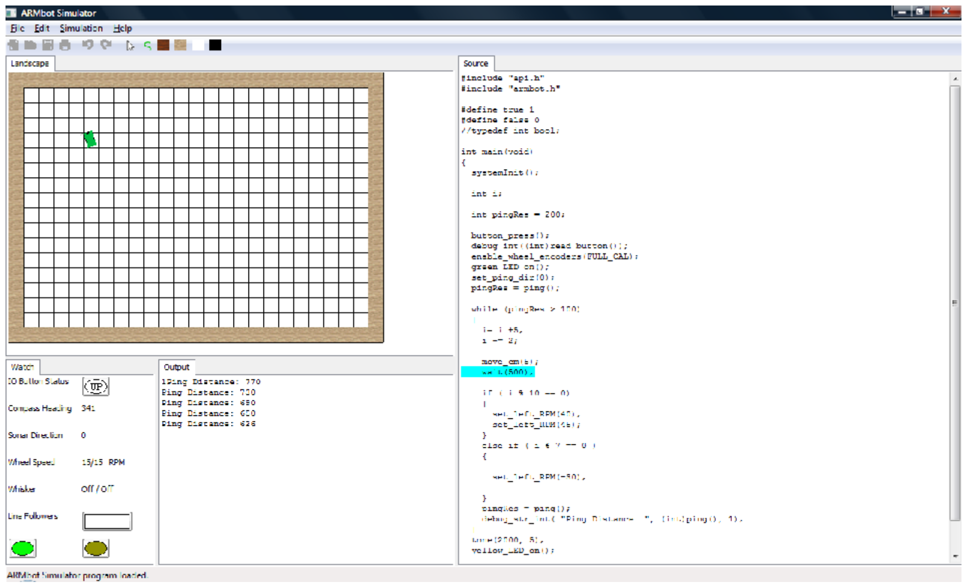
```
[08:51:19.296]: tok='Manager'
[08:51:19.296]: Namespace 'Manager'
[08:51:19.312]: tok='Get'
[08:51:19.312]: Looking for Get under Manager
[08:51:19.312]: Token found Get, type 'Manager'
[08:51:19.328]: actual type is Manager
[08:51:19.328]: Locating Manager
[08:51:19.328]: Class 'Manager'
[08:51:19.343]: tok=''
[08:51:19.343]: Scope='EditorManager'
[08:51:19.343]: Final parent: 'Manager' (count=28, search=)
[08:51:19.359]: Checking inheritance of Manager
[08:51:19.359]: - Has 0 ancestor(s)
```

Figure 2: Sample screenshot of the Code::Blocks development environment.

4 ARMbot Simulator

The ARMbot Simulator project provides a virtual testbed for robotic code, allowing users to observe robot behavior without needing to transfer executable code to an actual robot. Users can lay out black-line tracks which the robot's optical sensors can detect, and some obstacles for the robot to avoid, then place the robot at a starting position and orientation, and execute their code and watch the robot's movement.

The ARMbot Simulator uses many different utilities internally to accomplish this, including Perl scripts, a bundled compiler and preprocessor, etc. To properly incorporate the simulator into this project, the utilities that Code::Blocks uses for these common tasks are used instead. The project uses a separate frame to display code, and has its own code import methods; the corresponding Code::Blocks plugin instead reads code directly from an open file.



```
#include "api.h"
#include "armbot.h"

#define true 1
#define false 0
//typedef int bool;

int main(void)
{
  systemInit();

  int i;

  int pingRes = 200;

  button_press();
  debug int((int)read button());
  enable_wheel_encoders(FULL_CAL);
  green LED on();
  set_ping_dir(0);
  pingRes = ping();

  while (pingRes > 150)
  {
    i= i +5,
    i -= 2;

    move_cm(5);
    wait(500),

    if ( i % 10 == 0)
    {
      set_left_RPM(45),
      set_left_RPM(45);
    }
    else if ( i % 7 == 0 )
    {

      set_left_RPM(-50),

    }
    pingRes = ping();
    debug_str_int( "Ping Distance: ", (int)ping(), 1).
  }
  tone(2500, 5),
  yellow_LED_on();
```

IO Button Status
Compass Heading 341
Sonar Direction 0
Wheel Speed 15/15 RPM
Whisker Off / Off
Line Followers

```
1Ping Distance: 770
Ping Distance: 730
Ping Distance: 690
Ping Distance: 650
Ping Distance: 626
```

ARMbot Simulator program loaded.

Figure 3: Screenshot of the ARMbot Simulator.

5 ARMbot GUI

The ARMbot GUI is targeted primarily to elementary school students and is the focus of this project. With the ARMbot GUI, writing robot code is as simple as dragging and dropping icons onto the screen, which represent simple control structures (branching, looping, etc.) as well as robot function calls (check for distance to the wall, sound a bell, etc.) and other essentials. These code diagrams are converted to C code, which can then be simulated with the ARMbot Simulator, or compiled and loaded onto an actual ARMbot.

This project extends the ARMbot GUI's functionality by allowing users to save and load diagrams, or import preexisting C code as a new diagram, and extends the icon set to generalize the code generation to allow any API - not just the robotic toolkit - to be represented as icons.

When importing C code as a diagram, the code is first preprocessed using gcc to remove macros, and then parsed using lex/yacc and a custom C89 grammar to produce XML output, which outlines the code's functions, control structures, variables, function calls, etc. This same format is used to save and load diagrams.

6 Conclusion

This project brings together two ARMbot utilities under one environment, making them more accessible to the target audience, and adds additional functionality. This development suite will be used as an educational and promotional tool to attract interest for the field of robotics.

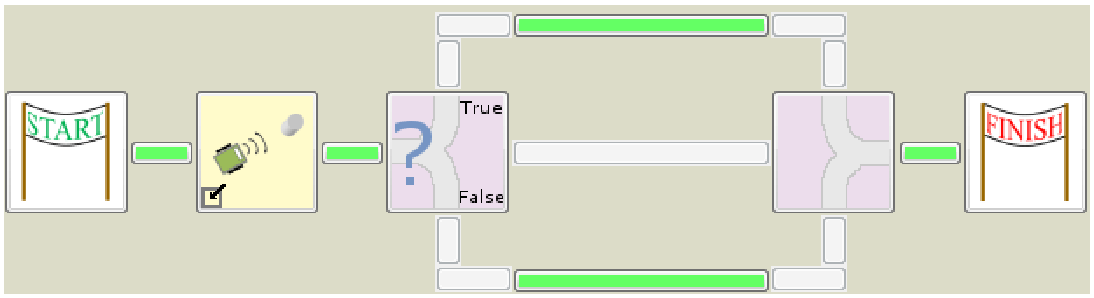


Figure 4: Example of a code diagram produced using the ARMbot GUI.