Educational robots for teaching programming to youths

Adam Bowes

Date

# Abstract

 **What you set out to do and why**

 **How you did it**

 **What you found**

Do this last

# Table of Contents

[Abstract 2](#_Toc413512945)

[Table of Contents 3](#_Toc413512946)

[Aims and Objectives 4](#_Toc413512947)

[Literature Review 5](#_Toc413512948)

[Methodology 6](#_Toc413512949)

[Evaluation 7](#_Toc413512950)

[Critical Reflection 8](#_Toc413512951)

[Appendices 9](#_Toc413512952)

# Aims and Objectives

Note - Not sure if this is needed or the format.

Robots have been used to further education and to increase engagement in a range of topics. For example, in secondary school mathematics robots have been used to demonstrate geometric transformations. Programming is a skill with which many have encountered difficulties understanding and persisting with, one cause is a lack of motivation and enthusiasm towards the topic. Another cause is that the interaction with the computer is limited to the computers screen preventing any physical or real world feedback. In this research I have created an interface to allow the use of Scratch, a user friendly programming language to control a Thymio-II robot to allow users to have their programs affect the real world.

In order to achieve this I have completed several objectives.

# Literature Review

Programming and computer skills are becoming increasingly important as the influence of the internet and the power of computers grows, programming has even been called the 'second literacy'. Despite this there are still significant barriers to education in this field and few attempts to integrate it in to other computer related topics. This project will use the language Scratch and the Thymio II as they have both been shown to be good at introducing people to the topic of programming and robots as well as maintaining interest and creating enthusiasm.

Scratch is a language developed at MIT which has been used in education with broad success. Scratch allows for the use of most programming concept without requiring the user to be aware of syntax through the use of blocks. As well as being easy to use it is also free to use and has a large community with a wide range of users from 4 to 60 year olds meaning support can be found relating to the most basic of tasks to complex ones. One study found that during a Harvard Summer School for Computer Science course that 76% of students felt that using Scratch as an introduction help them when they later moved on to java, students also found it was more rewarding to have visual feedback on what they had programmed than just having a text window (Malan and Leitner, 2007).

Amongst the reasons why teachers don't accept technology in to the class as readily as they do other tool are stress and fear of failure. When a teacher tries to teach using methods in which they have little experience they often find that it can be daunting and cause stress. Introducing people to Scratch has both caused people to be more likely to include programming in lessons and to worry about failing less when they consider programming courses or sessions. A study of students learning to become preschool teachers were given an introductory lesson in computer programming and found that interest in using technology in the classroom increased from 80% to 92%. As well as this they found that 65% found Scratch easy to use and 85% found it simple and understandable (Fesakis and Kiriaki, 2009).

One challenge facing robotics in education is the price of the platforms and how easy they are to use. The Thymio II can be bought for around £100 which is cheaper than alternatives such as the LEGO Mindstorm while still having most of the feature. Besides the LEGO robot there are few available robotics platforms that are simple enough that they can be used for an introduction to the topic while also being capable enough that they can perform complex programs. The Thymio II is a powerful system and with Scratch it would mean that it can be easily picked up by beginners while still being able to perform some complicated programs. Scratch include features such as the ability to create object orientate programs as well as use multithreading, and is seen to have one major limitation which is recursion which has been purposely left out so that beginners would not feel threatened by the complexity (Harvey, B. 2010).

Robots have been used before with other aspects of programming to create courses that have proven to create very enthusiastic students. For example, at the University of Lincoln robotics was taught alongside computer vision, this lead to positive results in practical sessions including some students ended up going far beyond the brief of their assignments with some advanced feature that they researched and implemented under their own direction. (Cielniak, G. et al, 2013), this suggests that enthusiasm can be created with practical assignments using robots.

The Thymio II is a programmable robot with a wide variety of sensors and methods for feedback. It has 2 wheels for movement, a speaker for audio output and several lights, some of which are programmable and others which indicate the feedback from the distance sensors. There are 9 distance sensors to prevent it from falling off objects and to detect thing in front or behind it. It also has a 3 axis accelerometer, a microphone and an infrared sensor for remote input. With these features the Thymio II is well suited to education as it can be applied to a lot of situations. The Thymio II is the result of testing amongst children with the Thymio II. After running courses with the Thymio 89.2% of parents thought the session was educational and 78.5% thought that it had increased their child's interest in robotics (Riedo, F. et al 2012)

### Conclusion

Several conclusions can be drawn from this research. Firstly, that robots used in education can lead to increased practical achievement and can motivate students to go beyond the constraints of an assignment. Secondly, that Scratch is a very powerful but simple language with a majority of standard features while still remaining user friendly and accessible to beginner programmers. As a result of this it can be used to introduce a variety of people to programming and to reduce their anxiety and discomfort with programming. Thirdly, that the Thymio II is the second iteration of a robot produced through extensive user feedback and as a result is very capable and adaptable platform.

# Methodology

## Project management

Agile/learn/extreme

## Software Development

-software development methodology  
-characteristics of the software and the computer environments available.  
-simplistic review of strengths and weaknesses

**Needs references? - yes**

This project required a piece of software to be developed. This software took form as an interface allowing communications between the programming language Scratch 1.4 and the Thymio-II robot. To create this interface the incremental software development method was applied. Incremental development is a variation on the waterfall method which consists of multiple waterfalls with reviews between each allowing waterfall to be used on flexible projects.

This method was chosen after considering the characteristics of the software being built and the environment in which it was constructed. Developing the interface required overcoming unique challenges which were largely unique to this type of software, as a result some segments of the code needed to be rewritten and upgraded as new knowledge was gained. Incremental development allows for the software to be reviewed after each task is completed so that improvements to the code can added as tasks at suitable points in the development. Consistently improved code was important for this project as this meant the scope of the interface could expand without causing conflicts with obsolete code. Scratch 1.4 and the Thymio-II robot both needed to communicate with the interface. As neither components were built with compatibilities for the other the development process had to account for impassable obstacles which could result in fundamental aspects of the interface needing to be changed such as the programming language or the methods of communication. Incremental development suited this as its review process gives an opportunity to evaluate the current state of the program in addition allowing newly acquired knowledge to be applied to the situation.

The result of using this method meant that during initial development when experimentation took place there was little commitment to carrying on with a particular plan. Following this unsuccessful ideas could be discarded quickly and so that the core functionality of the interface was more robust. The incremental aspect of the development meant that the software was completed in discrete and complete chunks such that each iteration resulted in a working prototype with individually changing features. This discrete improvement meant that progress could be monitored easily by reviewing which features had been added or changed. In addition to this the scope of the interface was expand where appropriate during the brief review periods to include feature which previously could not be planned for as the relevant knowledge about the system was not known.

One issue with the method was that features were often completed on their own without consideration for future iterations. This caused large amounts of the interface to need to be rewritten, although this had the advantage of ensuring the program was built with recently acquired knowledge it also caused the progress of the project to slow down. With a more rigid method features could be anticipated and prepared for better. Another issue with the development method was that while it was useful for increasing the scope of the software and monitoring progress it wasn’t very useful for estimating the tasks remaining and the time they would take. As the software needed to be functional before any testing or studies could take place

Cms, (2005) *Selecting Developing Approach*. [Online] Available from: www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf [Accessed 18 October 2014].

# Design

## Introduction

In this section the components of the interface will be broken down and the external communication options will be analysed. After this the design of the Interface will be presented.

For this project an interface between Scratch 1.4 and the Thymio-II robot was developed. In order to build this the architecture and functionality of each component had to be evaluated and designed for. The Scratch 1.4 programming language contains trigger blocks which react to events allowing for event based programs to be developed. On top of this Scratch 1.4 supports concurrent processes and thread based programming (Maloney, J. et al, 2010). In relation to building an interface this flexibility allows for a range of approaches to be made. The Thymio-II makes use of its own programming language where code is executed when events are triggered. These events include timers, sensor updates and user created functions (Shin, J., and Magnenat, S. 2014). This event based programming caused conflicts with thread based programs in Scratch 1.4. The interface was decided to be built using python which supports multithreading as well as event based programming.

## Scratch 1.4 communication

Internally Scratch 1.4 uses broadcasts which sends a message to all objects in the project, if an object contains a trigger for that specific message then it will be executed. As well as broadcast the variables in Scratch 1.4 are global allowing objects to communicate values directly.

Scratch 1.4 already has support for external sensors in the form of remote sensor connections (RSC). Enabling the RSC causes scratch to run a server which will transmit all variable changes, variable creation and broadcast to all connected devices. Broadcast are sent as a string containing the value of the message being broadcast. Variable changes and creation are sent with the variable name and the new value. Unlike standard variables in Scratch 1.4 variable sent over the RSC will be received as sensor values which cannot be edited. Broadcasts can also be received by the RSC server which will broadcast the message to all objects in the Scratch project. Variables can set and created by send the RSC server the variable name and its value.

The ability for variables to be created through the RSC server allows for the interface to send any value without Scratch or the interface encountering a fatal error. Sensor values is the main way for data to be communicated in to Scratch 1.4 through the RSC server, as the value can’t be edited it also won’t be transmitted back to the interface. Due to the inability to easily edit standard variables in Scratch 1.4 from the interface feedback to the user about parameters entered cannot be easily given.

## Thymio-II communication

Besides the custom built software the Thymio-II supports communication over D-Bus, a proprietary Linux protocol for communication. Using D-Bus variables can be requested. Requesting variables requires a handler to manager the different variable type that can be received. Variables on the Thymio-II can also be adjusted. The Thymio-II changes its state based of its local variable such that a variable corresponding to a LED can be altered to change the brightness of that LED.

Receiving messages from the Thymio-II requires a command to be sent first and then a response containing data to be sent back. Requesting large amounts of data often can cause network latency to increase and prevent other messages from being received. Sending commands to the Thymio-II requires variables to be changed, after this the Thymio-II will then have to recognise these changes and then relay the changes to the actuators. This method of sending commands is simple and handles timing and optimisation automatically, however as the Thymio-II might not react instantly some commands might be delayed.

# Interface

[Diagram]

# Development

## Introduction

This section will explain the development process of the project and will go over the steps taken to create and use the interface. In addition any issues encountered will be described along with their solutions. The interface will be explained first followed by the theory behind connecting the interface to the robot.

## Interface implementation

The first task that had to be undertaken was enabling Python to communicate to the Thymio. After some research it was clear that Asebamedulla could be used to allow a D-Bus connection to communicate with the Thymio-II. Using Pythons D-Bus module the code could connect with the Thymio-II via Asebamedulla using a D-Bus Interface and the Asebamedulla D-Bus name (figure [X]).

Figure 1: The D-Bus network being created to asebamedulla

Now there was a connection to Asebamedulla which would then send our data through to the Thymio-II robot which would apply them as appropriate. The Python code now needs to retrieve and store the current values from the Thymio. To do this a loop was created that would send a command every 100 milliseconds telling the Thymio to send the current values from its proximity sensors. The sensor data could be accessed using the previously declared network variable along with the GetVariable command. The GetVariable command required several parameters to function. Firstly it needs the device to be specified which in this case is the Thymio-II. Next the variable that is meant to be returned is set, for the proximity sensor the string “prox.horizontal” is sent. The next parameter is a function to handle the data and finally there needs to be a function to handle any errors. The variable handling functions make use of variables declared with appropriate types meaning that for each sensor type the data can be stored directly in to a variable. The error handling receives an error message in the form of a string and cause the current code to quit (figure [X]).

Figure 2: The variable handling for receiving a variable

Due to the nature of Python variables can be declared with a type and then referred to outside of their initial scope by using the key word “global”. The proxSensorVal holds an array of 7 floating point variables where each is the current distance the corresponding proximity sensor. The groundDeltaVal stores 2 values for the downward facing proximity sensors which are used to tell if there is a surface below the front of the robot. The accVal stores 3 floating point variables, each of which representing an axis from the Thymio-II’s accelerometer.

Before the interface and send commands the code first needs a way of dealing with user input. An array of strings is used where each string is the text for a command (e.g. “forward”). This array is used to set a single string to a variable called command which stores the current command. As the loop progresses a series of if statements check for the different commands and runs the appropriate command.

The Thymio-II deals with movement by setting a target speed for each wheel and then automatically tries to achieve this. The interface can take advantage of this by sending a command to set the target speed variables to the required speed instead of trying to access functions on the Thymio-II. The command to set the variables makes use of the network object and is called SetVariable. This function is similar to the GetVariable function in that it requires the device and the variable name but then you send the value you want to set the motor to (figure [X]).

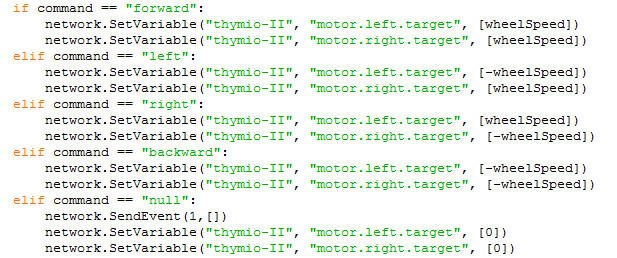


Figure 3: Setting wheel target speed depending on command

Now that data has been received and commands can be sent the interface now needs to communicate with Scratch 1.4. Scratch 1.4 has a built in server for robot communication which allows values to be read in. On top of this, with the server enabled Scratch 1.4 will send all broadcasts and variable changes over the network. To connect to this the Python code will make use of a python module called scratch. This module connects to Scratch 1.4 and handles all data traveling between the interface and Scratch 1.4. To initialise the connection to Scratch 1.4 a Scratch object is made, during this the host IP address is given (the address to the machine running scratch) and the module establishes the link. At this point any broadcast made in Scratch 1.4 will be sent and added to the back of the received commands queue.

As the main loop needs to keep running to update sensor data

1 connect to Thymio -  
2 receive data -  
3 send commands -  
4 connect to scratch -  
5 Receive commands from scratch  
6 transfer scratch commands into Thymio commands  
7 transfer thymio data to scratch  
8

# Evaluation

# Critical Reflection

# Appendices

Riano, L. and McGinnity ,T. (2010) Design and validation of a robotic system to interactively teach geometry, *AT&P Journal* (*2)*  91-96

Cms, (2005) *Selecting Developing Approach*. [Online] Available from: www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf [Accessed 18 October 2014].

Malan, J., and Leitner, H. (2007) Scratch for Budding Computer Scientists, *ACM SIGCSE Bulletin* 39(1) 223-227.

Fesakis, G., and Kiriaki, S. (2009) Influence of the familiarization with Scratch on future teachers' opinions and attitudes about programming and ICT in education, *ACM SIGCSE Bulletin*, 41 (3) 258-262.

Harvey, B., and Mönig, J. (2010). Bringing “No ceiling” to Scratch: can one language serve kids and computer scientists, *Proc. Constructionism*.

Cielniak, G., Bellotto, N. and Duckett, T. (2013) Integrating mobile robotics and vision with undergraduate computer science, *Education, IEEE Transactions on* 56(1) 48-53.

Maloney, J., Resnick, M., Rusk, N.,Silverman, B. and Eastmond, E. (2010) The scratch programming language and environment, *ACM Transactions on Computing Education (TOCE)* 10 (4) 16.

Shin, J., and Magnenat, S. (2014). *Visual programming language for Thymio II robot*. ETH-Zürich.