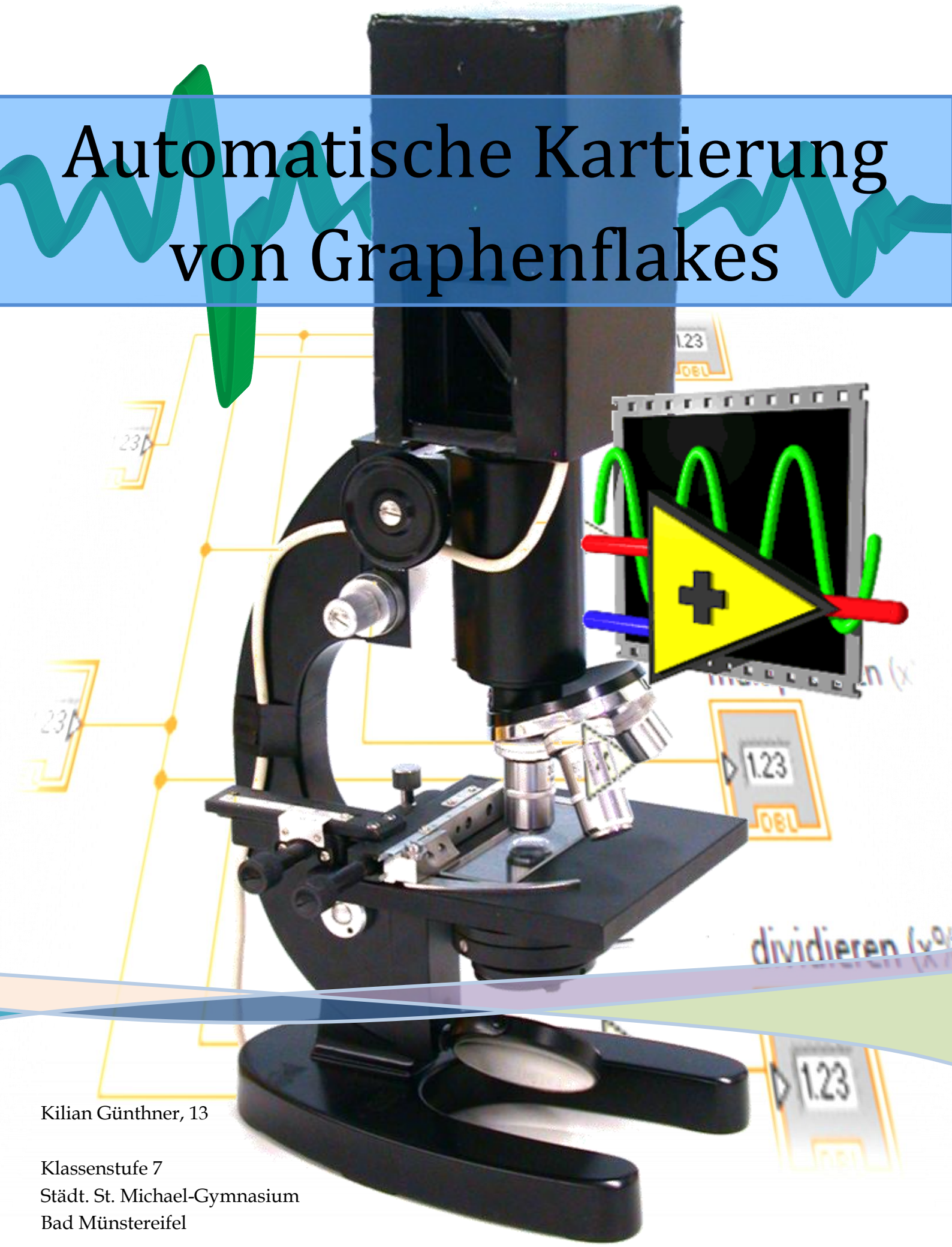
Automatische Kartierung von Graphenflakes



Kilian Günthner, 13

Klassenstufe 7
Städt. St. Michael-Gymnasium
Bad Münstereifel

Gliederung:

Inhaltsverzeichnis

1. Kurzfassung

Letztes Jahr beschäftigte ich mich mit dem Bau eines digitalen Auflichtmikroskops, welches allerdings noch verbesserungswürdig war: es dauert sehr lange, um auf einem SiSiO_2-Wafer Graphen zu finden. Eine andere Jugend-forscht-Gruppe hat zurzeit dieselben Probleme.

Deshalb beschloss ich, mich damit zu beschäftigen ein Programm zu schreiben, dass die Graphenflakes auf der Probe automatisch finden und auf einer digitalen „Landkarte“ der Probe hervorheben soll. Da ich das Modell möglichst kostengünstig verwirklichen wollte, begann ich mit Lego Mindstorms, ein Baukasten zum Roboterbauen und -programmieren. Ich baute an ein Mikroskop eine Vorrichtung mit Lichtsensor und zwei Motoren, konnte aber leider nur bei Proben mit großen Kontrastunterschieden „Graphen“ wahrnehmen. Daher recherchierte ich nach einem, für meine Zwecke besser geeignetes Programm. Ich stieß dabei auf LabVIEW, eine einfach-visuell aufgebautes Programmiersoftware zu der mir die Firma National Instruments freundlicherweise das Vision-Tool zur Verfügung stellte. Dieses Tool stellte mir dann auch die Möglichkeit zur Verfügung Bilder zu erkennen und digital zu verarbeiten. Momentan ist meine Bilderkennung auf dem Stand, dass ich auf einem Standbild nach einer festgelegten Form oder Farbe Graphen erkennen kann.

2. Idee und Zielsetzung

Im letzten Jahr beschäftigte ich mich mit der Entwicklung eines digitalen Auflichtmikroskops. Ich erzielte zwar schon sehr gute Ergebnisse (siehe Abb. 2.1) musste mir aber immer, bevor ich Graphen oder anderes zeigen wollte, mindestens eine dreiviertel Stunde Zeit reservieren, da ich erst das Graphen auf dem, für diese Vergrößerung „riesigen Platz“ suchen musste. Die Jugend-forscht-Gruppe „Graphen“ hat zurzeit auch ähnliche Probleme. Da wir uns natürlich alle mit unseren Projekten beeilen sollten und dies auch in Zukunft müssen, nahm ich mir vor, ein Programm zu entwickeln, das die langen Suchzeiten verringern sollte, indem es die Probe komplett abrastert, Unregelmäßigkeiten erkennt und diese dann auf die optischen Eigenschaften von Graphen untersucht. Wenn dieser Vorgang abgeschlossen ist, soll

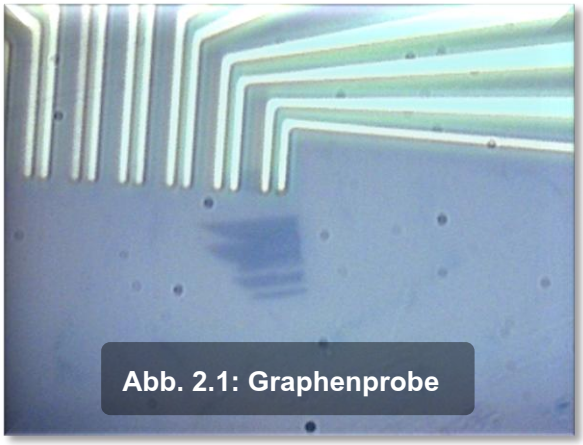


Abb. 2.1: Graphenprobe

das Programm eine interaktive „Landkarte“ erstellen, welche die Bilder, die bei der Abrasterung entstanden, zu einem Bild zusammenfügt, über die sich dann die im Programm farblich hervorgehobenen Felder mit Graphen anfahren lassen.

3. Lego Mindstorms

Ich habe überlegt, wie ich das Modell am einfachsten und preisgünstigsten verwirklichen könnte. Dabei erinnerte ich mich, wie ich vor einigen Jahren Roboter gebaut habe: mit Lego Mindstorms. Lego Mindstorms ist ein von Lego entwickeltes Paket, welches einen intelligenten Hauptstein, den „RCX“ beinhaltet, sowie die Standardsensoren und -motoren.
Der Hauptstein lässt sich über eine spezielle Programmiersoftware extra für dieses Produkt kontaktieren. Über jeweils zwei Legosteine mit Elektroden, die mit einem Kabel verbunden sind, lässt sich der RCX mit den Sensoren und Motoren verbinden. Der Hauptstein selbst lässt sich über eine Infrarotstation kontaktieren.

Mein erster Versuch mit Lego Mindstorms sollte darin bestehen, den Lichtsensor in eine Form einzuspannen (Abb. 3.3), welche den Sensor über ein Blatt mit einer schwarzen Linie bewegen sollte. Der Sensor sollte, wenn er diese erblickt, die Motoren anhalten und ein Warngeräusch von sich geben. Danach wollte ich (wie es dann ja auch im Endmodell sein wird) nicht die „Kamera“ als Lichtsensor, sondern die Probe selbst beweglich machen.
Ich startete den Versuch, indem ich an den, in den X und Y Achsen verstellbaren Objekttisch, zwei Motoren montierte, welche sich über den RCX bewegen ließen. Den Lichtsensor montierte ich am Objektiv des Mikroskops.
Um die Software zu programmieren, benutze ich die von Lego extra für dieses Produkt entwickelte Software (Softwareanatomie: Abb. 3.1 und 3.2). Als erstes führte ich den Versuch mit einer schwarz-weißen Vorlage durch.
Das funktionierte jetzt zwar schon gut, war aber noch nicht was ich wollte.
Daher erstellte ich eine Zeichnung mit den ungefähren Kontrastunterschieden, die Graphen zum Hintergrund der Probe auch aufweist.
Leider funktionierte das nicht sehr gut. Also recherchierte ich im Internet nach einem professionellen Programm mit dem man einfach komplexe Programme schreiben kann. Ich stieß auf die Internetseite der Firma „National Instruments“, welche genau die Software produziert, die ich benötigte. Die Software trägt den Namen „LabVIEW“.

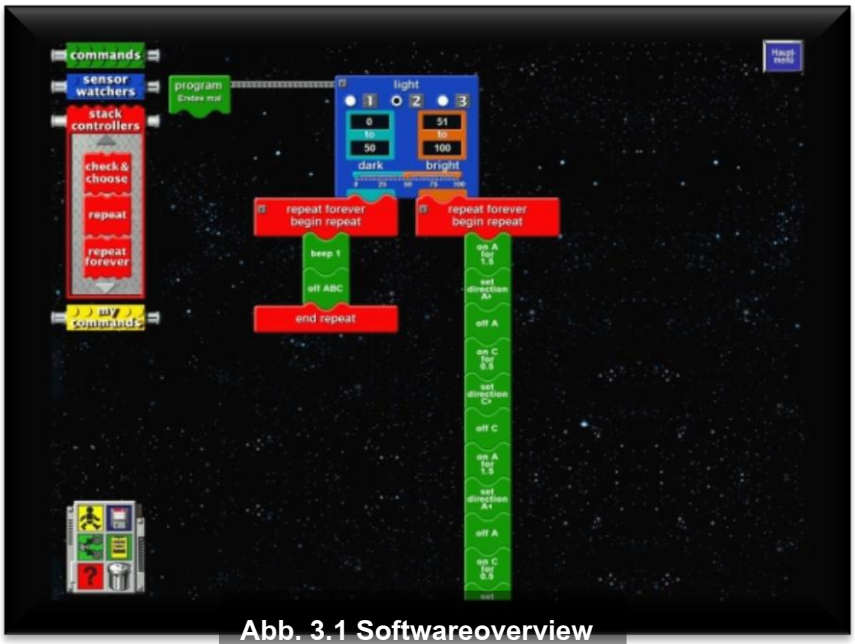


Abb. 3.1 Softwareoverview

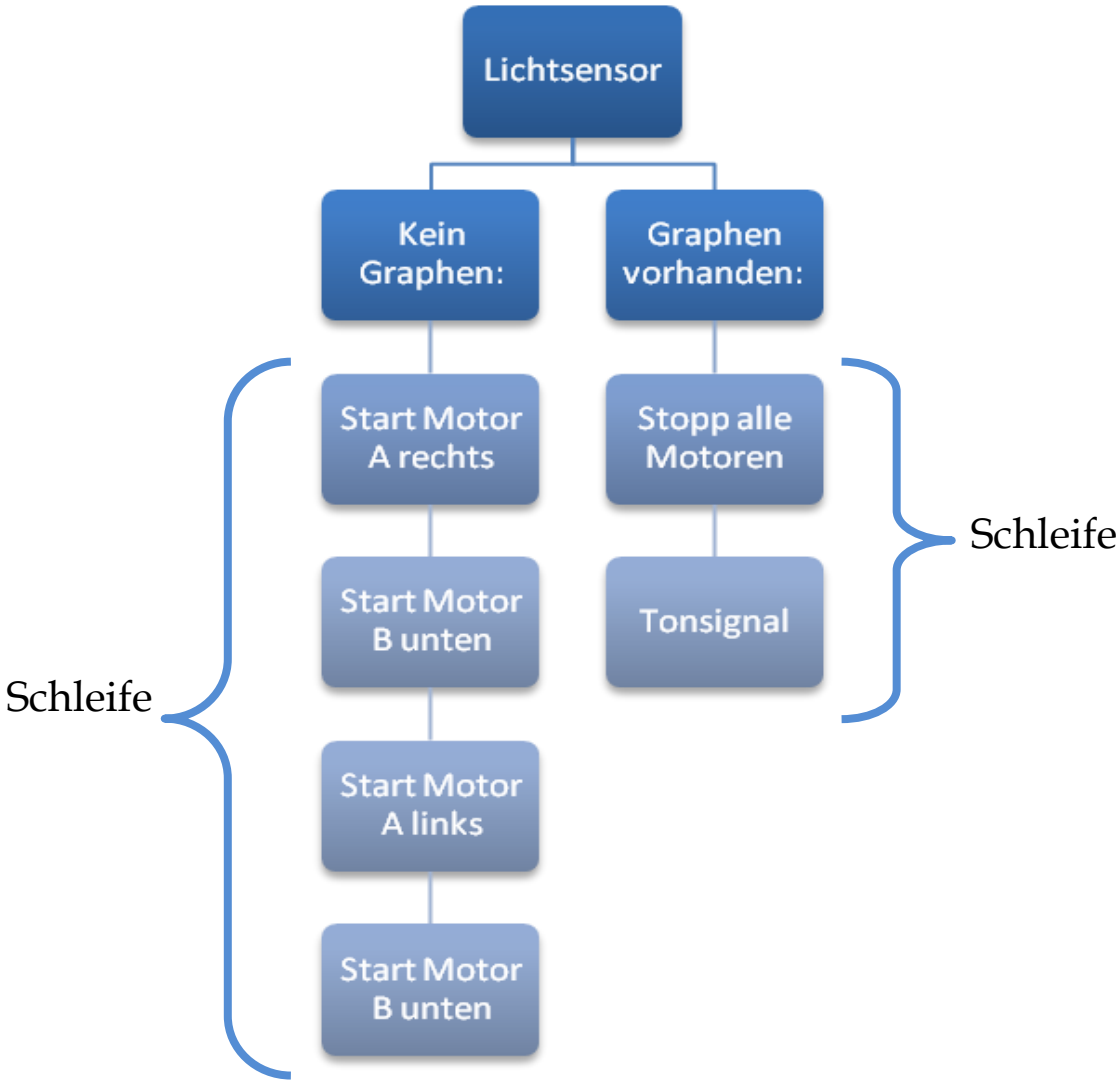


Abb. 3.2: Softwareanatomie

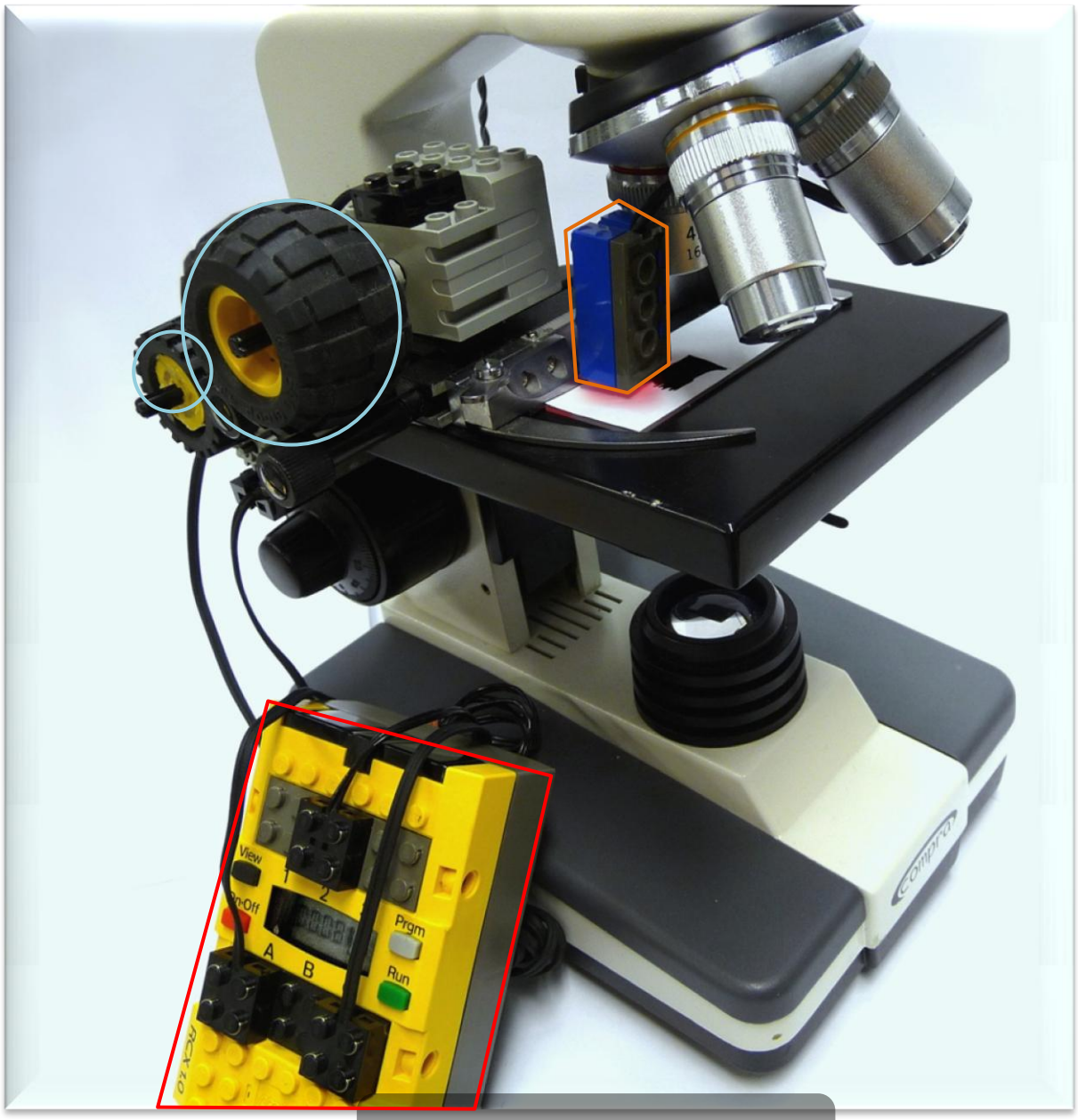


Abb. 3.2: Erste Konstruktion mit Lego Mindstorms

= Hauptstein (RCX)

= Motoren (X und Y)

= Lichtsensor

4. LabVIEW und Vision

4.1 Allgemeines

LabVIEW ist ein von **National Instruments** entwickeltes Programm, das zur Programmierung von Mess- und Laborprogrammen mit einfacher Visualisierung geschrieben wurde. Es lässt sich sehr leicht bedienen, da es zwei Hauptfenster gibt. Einmal das **Frontpanel**, welches zur Gestaltung der späteren Benutzeroberfläche gedacht ist. Außerdem das **Blockdiagram**, in welchem die, in der Benutzeroberfläche enthaltenen Objekte als kleine, viereckige Symbole dargestellt werden. Zwischen diese lassen sich die sogenannten Funktionen ziehen, die meistens über mindestens einen Ein- und Ausgang verfügen. Objekte aus dem Frontpanel lassen sich mit Funktionen mit Hilfe von „Drähten“ verbinden. So lässt sich ein Programm ohne Kenntnisse einer bestimmten Programmiersprache verwirklichen. Man kann zum Beispiel einen Taschenrechner, der zwei Zahlen gleichzeitig addiert, subtrahiert, multipliziert und dividiert ganz einfach innerhalb von höchstens drei Minuten programmieren. Man muss einfach dem Frontpanel zwei numerische Eingabefelder, sowie vier numerische Anzeigefelder hinzufügen. Beide kann man beliebig benennen. Im Blockdiagramm zieht man einfach vor jedes Anzeigefeld eine numerische Funktion, die den entsprechenden Titel trägt (z.B.: „+ - * /“). Diese Funktionen haben schon zwei Eingänge und einen Ausgang. Nun „verkabelt“ man die Ausgänge der numerischen Eingabefelder mit jeweils einem Eingang einer Funktion und die Ausgänge der Funktionen mit den Anzeigefeldern (siehe Abb.4.1.1).

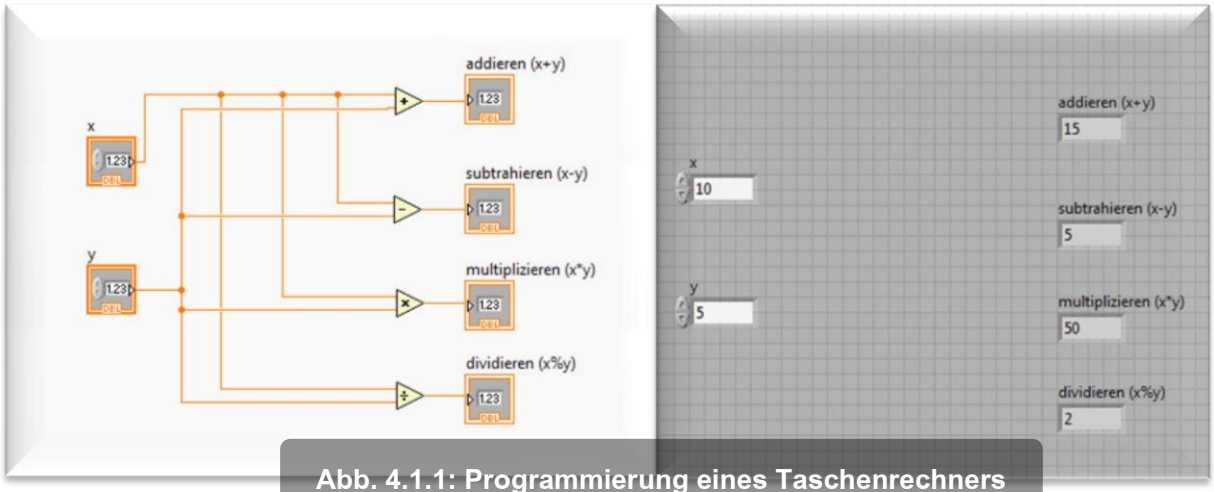


Abb. 4.1.1: Programmierung eines Taschenrechners

4.2 Graphenerkennung

Ich benutzte den von NI (National Instruments) beigelegten „**Vision Assistent**“, um mit der Graphenerkennung zu beginnen. Dazu begann ich erst einmal einen Testlauf mit der **Formerkennung**, indem ich das Bild von einem Taschenrechner nahm und die **ROI** (zu Deutsch: Region von Interesse) um das Display legte. Das Programm erkannte die verschiedenen wichtigen Formen, welche ich dann nur noch zur Wiedererkennung benennen musste (siehe Abb. 4.2.1). Das Prinzip der Formerkennung konnte ich nun auch auf ein Standbild einer Graphenprobe anwenden (siehe Abb. 4.2.2). Ich legte die ROI um die Graphenprobe, das Programm erkannte das **IO** (Important Objekt) und ich musste es zur Wiedererkennung nur benennen. Das Problem bestand aber nun darin, dass ich immer noch die ROI selbst festlegen musste und nicht einfach das ganze Bild als solche verwenden konnte, da ansonsten noch sehr viel mehr als IO erkannt werden würde. Außerdem hat Graphen nicht immer die Form, die auf dem Bild vorhanden war, und so hätte ich am Ende noch mehr Arbeit als zuvor. Zwar schon mal ein Schritt in die richtige Richtung, der mich aber nicht voran brachte.

Daher probierte ich es mit der **Farberkennung**. Da diese weitaus schwieriger zu verwirklichen ist als die Formerkennung, malte ich ein einfaches Bild mit dem Programm „Paint“. Diese Zeichnung sah (mehr oder weniger) einer Graphenprobe ähnlich, hatte aber zu meinem Vorteil keine leichten oder schweren Verfärbungen. Um festzustellen, wo sich das Graphen befindet, ist es erst einmal wichtig, dass man dem Programm sagt, welche Farbe das Graphen hat: man lässt das Programm ein Foto von der favorisierten Farbe schießen. Dieses Foto wird danach Pixel für Pixel auf jeden Teil des Ursprungsbildes gelegt und beide werden verglichen. Stimmen die Farbwerte nun großteils miteinander überein (die Empfindlichkeit ist verstellbar), markiert das Programm dieses Feld auch und hebt es rot hervor, sodass an den Stellen, an denen sich Graphen befindet, am Ende Sammlungen roter Felder zu sehen sind (Abb. 4.2.3).

Abb. 4.2.1: Taschenrechner

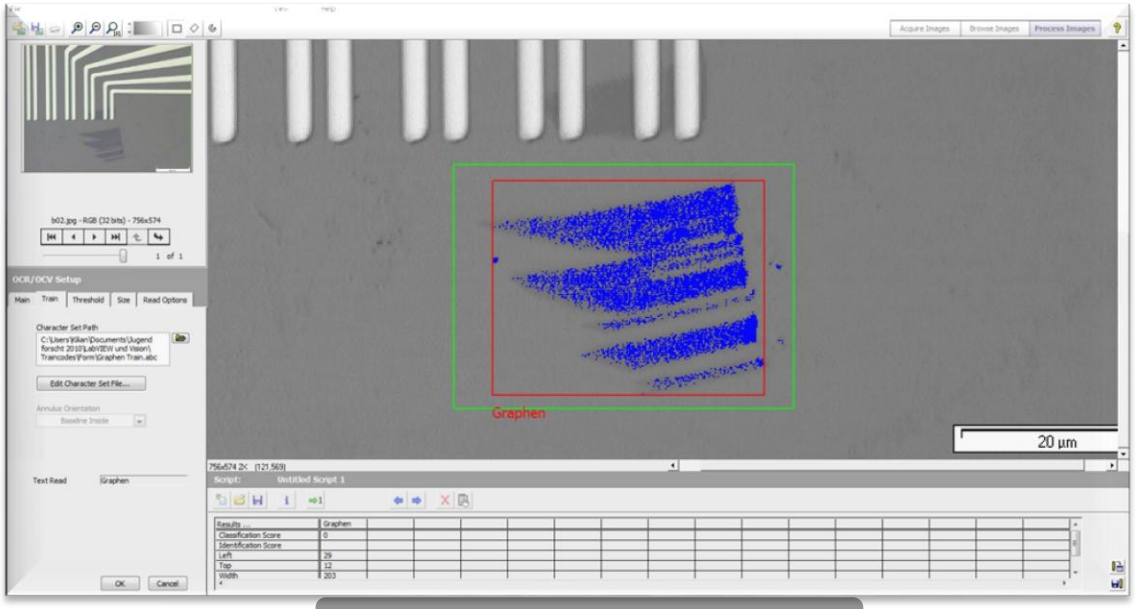


Abb. 4.2.2: Graphen Formerkennung

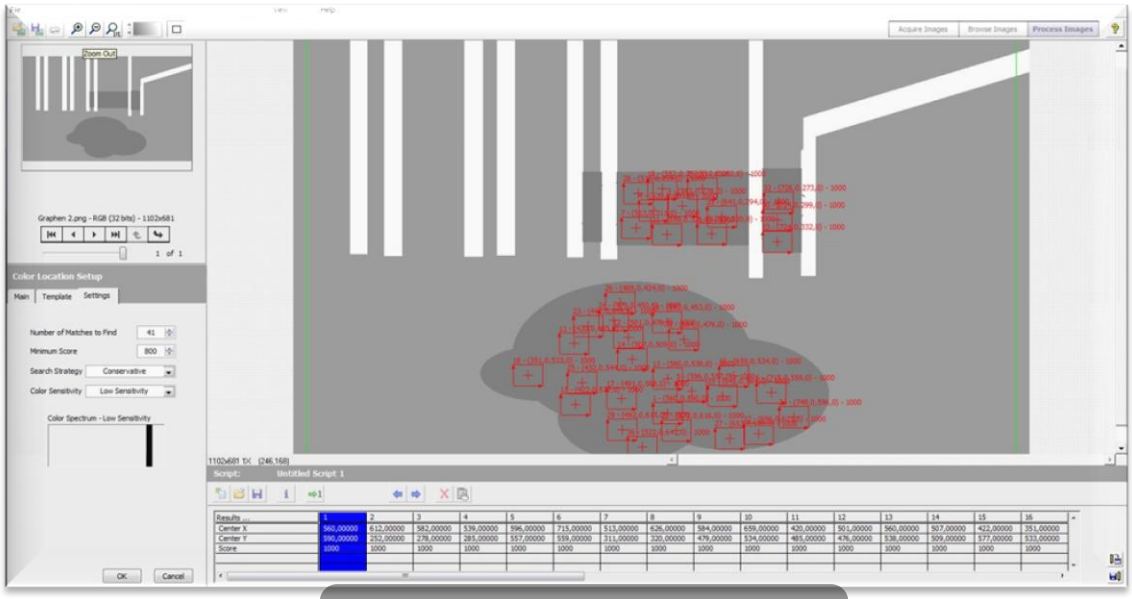


Abb. 4.2.3: Gezeichnetes und von Vision erkanntes Graphen

5. Ausblick

Insgesamt habe ich während meines Projektes schon sehr große Fortschritte gemacht und schon einen großen Teil meiner Ziele erreicht. Es gibt aber immer noch Dinge die ich noch nicht erreicht habe. Zum Beispiel möchte ich die Motorsteuerung selbst noch mit LabVIEW verbinden. Außerdem möchte ich die Motorsteuerung (bisher Lego Mindstorms) durch einen Objekttisch in den bereits Motoren integriert sind ersetzten. Ebenso möchte ich den Lichtsensor, den ich bisher an dem Objektiv befestigte, durch den CCD-Chip zu ersetzen, welcher dann aber nicht wie zuvor der Lichtsensor am Objektiv befestigt werden soll, sondern die altbekannte Position über dem halbdurchlässigen Spiegel beanspruchen wird.
Das Mikroskop selbst soll am Ende auch einige Änderungen haben.
Das heißt, dass ich die Lampe an das Mikroskop anbinden bzw. integrieren möchte.
Außerdem möchte ich den Rest des Mikroskops auch noch stabilisieren.

6. Danksagung

Ich möchte ganz herzlich der Firma NI (National Instruments) danken, die mir freundlicherweise das Vision Tool zur Verfügung stellte und mir somit überhaupt ermöglichte das meine Ziele in erreichbare Nähe rückten. Hier noch einmal vielen Dank.

7. Literatur- und Linkliste

http://www.et.byu.edu/groups/uolab/images/labview.gif

http://www.ni.com/labview/d/