Kooperative, verteilte Überwachung eines Gebiets auf Eindringlinge

Dennis Lisiecki, Torsten Kühl

I. Einleitung

Eine Überwachungskamera dient im allgemeinem dem Zweck, einen bestimmten Bereich dauerhaft zu überwachen. Das Bild wird kann direkt auf einem Monitor wiedergegeben und parallel aufgezeichnet werden, in Supermärkten erfolgt die Ausstrahlung des Live-Bilds sogar zur Abschreckung oftmals direkt im Verkaufsraum. Zum Schutz vor Vandalismus oder ähnlichem steigt auch im privaten Bereich die Verbreitung von Überwachungskameras. Dabei tendiert der Markt zu immer kleineren Modellen, mit höheren Auflösungen. Geräte solcher Art gibt es bereits zu Genüge, weswegen unser Projekt sich auf einen anderen Ansatz konzentriert.

Ziel war es, mithilfe von zwei oder mehr Rasperry Pis, die jeweils mit Kamera und Infrarot-Sensoren bestückt sind, ein Gebiet oder Raum auf unerwünschte Eindringlinge zu überprüfen. Die Geräte sollen dazu an unterschiedlichen Stellen platziert werden, die den zu überwachenden Raum aus unterschiedlichen Blickwinkeln beobachten. Die Kamera des jeweiligen Geräts soll nun auf jegliche wahrgenommene Bewegung reagieren. Um die Genauigkeit zu erhöhen und eine Aussage über die Art der erkannten Bewegung treffen zu können, soll die Kamera außerdem mit dem Infrarot-Sensor, (”PIR-Sensor”) kooperieren.

Der PIR-Sensor reagiert lediglich auf Objekte, die eine Wärmesignatur ausstrahlen. Wenn nun durch die Kamera und den PIR-Sensor eine Bewegung erkannt wurde, kann der Benutzer dies anhand der Handy-App nachvollziehen. Die Wahrscheinlichkeit, dass es sich um einen Mensch oder ein Tier handelt wäre in diesem Fall sehr hoch.

So weit unterscheidet sich das System, abgesehen von der App für Mobiltelefone als Informationszentrum, kaum von anderen Lösungen zur Überwachung. Um eine noch höhere Genauigkeit zu erzielen, sollen die Geräte bei unserem Ansatz allerdings zusätzlich noch miteinander kommunizieren:

Eine einzelne, lediglich durch die Kamera des Raspberry Pi wahrgenommene Bewegung stellt das geringste Risiko dar. Zwei oder mehr unterschiedliche Raspberry Pis, welche mit Kamera und PIR-Sensor eine Bewegung feststellen, stellen das höchste Risiko dar. Wie das funktioniert und in welcher Art und Weise die Systeme miteinander kommunizieren, wird im Folgendem erklärt.

II. System

A. Raspberry Pi

Der Raspberry Pi, welcher in unserem Projekt die Grundlage bildet, ist ein voll funktionsfähiger PC im Scheckkartenformat. In erster Linie wurde der Raspberry Pi mit dem Ziel entwickelt, interessierten Menschen das erlernen von hardwarenaher Programmierung zu erleichtern. Jedes Gerät besitzt ein frei programmierbares General Purpose Input Output-Board (”GPIO-Board”). Das GPIO-Board stellt je nach Modell 26 oder 40 Pins zur Verfügung, von denen 17 bzw. 26 Pins frei programmierbar sind und die weiteren der Spannungsversorgung oder als Masse dienen. Die einzelnen Pins dieses Boards lassen sich mit selbst programmierten Programmen ansteuern und können vielseitigen Zwecken dienen. So kann diverse externe Peripherie angesteuert werden, wie z.B. ein Temperatur-Sensor, ein Ultraschall-Sensor, ein Motor oder sogar ein kleiner externen Monitor mit Touch-Funktion. Als Betriebssystem können unter anderem an die Architektur angepasste Linux-Distributionen wie das auf Debian basierende *Raspbian* installiert werden. Auch Betriebssysteme, welche den Raspberry Pi zum Mediencenter umfunktionieren, um damit Filme und Musik abzuspielen, sind von der Community mittlerweile zur Verfügung gestellt worden. Wie solche Projekte zeigen, ist der Raspberry Pi nicht nur zum lernen gut geeignet. Die Video- und Audioausgabe erfolgt über eine HDMI-Schnittstelle, für die Audioausgabe steht alternativ auch ein 3,5mm Klinkenanschluss zur Verfügung. Für die Stromversorgung wird ein 5-V-Micro-USB-Anschluss genutzt. Hier stehen dem Anwender viele Türen offen:

Neben z.B. den meisten Handy-Ladegeräten kann die Stromversorgung auch über Batterie und Solarzelle erfolgen. So kann der Raspberry Pi auch mobil verwendet werden. Bis dato konnte sich der Raspberry Pi knapp 4 Millionen mal verkaufen und ist inzwischen in seiner vierten Version erschienen. [5] Die erste Version dieses Rechner kam Anfang 2012 auf den Markt und erfreut sich seither größter Beliebtheit. Je nach Ausführung ist das Gerät zwischen 25 und 35 Euro teuer. Die unterschiedlichen Ausführungen unterscheiden sich in gewissen Punkten:

Modell A und A+ besitzen 256 MB Arbeitsspeicher und nur einen USB-Anschluss, Modell B und B+ besitzen 512 MB Arbeitsspeicher, eine Ethernet-Schnittstelle sowie zwei, respektive vier USB-Anschlüsse. Alle Modelle müssen ohne Festplattenschnittstelle auskommen und verwenden SD-Karten bzw. Micro-SD-Karten als Speichermedium. Für unsere Ausarbeitung haben wir den Raspberry Pi B+ verwendet.

B. Raspberry Pi Kameramodul

Für unsere Ausarbeitung verwenden wir das Raspberry Pi Infrarot Kamera Modul (”Pi NoIR Camera Board”), weil es

vom Raspberry Pi selbst auf jeden Fall unterstützt wird und der Support der Raspberry Pi Community hervorragend ist. Die Kamera bietet eine Auflösung von bis zu 5 Megapixel und kann bei statischen Aufnahmen mit einer Auflösung von bis zu 2592 x 1944 Pixel aufwarten. Mit Abmessungen von 25 x 20 x 9 mm ist die Kamera äußerst klein, muss allerdings auch ohne eigenes Gehäuse auskommen. Für die Raspberry Pi Kamera gibt es am Raspberry Pi einen eigenen Slot, in dem das Flachbandkabel der Kamera passt, sodass die GPIO-Anschlüsse am Raspberry Pi vollständig für andere Aufgaben verwendet werden können. Da die Kamera dazu in der Lage ist, Licht aus dem Infraroten Spektralbereich einzufangen, kann man auch bei schlechter Beleuchtung oder nachts noch Bewegungen erkennen lassen oder Fotos schießen. Dazu muss bei Nacht der entsprechende Bereich mit Infrarotlicht ausgeleuchtet werden. Bei vielen Überwachungskameras wird dieses Licht mittels LEDs erzeugt, welche um das Objektiv der Kamera platziert werden. Für unser Projekt sollen allerdings vorerst keine Infrarot-LEDs zum Einsatz kommen. Natürlich können zur Bewegungserkennung auch gewöhnliche USB-Webcams verwendet werden. Wir geben allerdings zu bedenken, dass das Python Script, das wir zur Bewegungserkennung verwenden, nur das Raspberry Pi Kamera Modul unterstützt. Aber auch für diesen Anwendungsfall gibt es Lösungen, die wir für die Verwendung in unserem Projekt in Betracht gezogen haben. Dazu später mehr.

C. PIR-Sensor

Beim zweiten Sensor handelt sich um ein passiven Infrarot Sensor:
Der PIR-Sensor (Passive Infrared Sensor) ist einer der gängigsten Bewegungsmelder und ist oftmals auch in bewegungssensitiven Außenleuchten oder Alarmanlagen verbaut. Erkennbar ist der PIR-Sensor an seiner meist runden, milchigen Kuppel, die mit vielen einzelnen Waben versehen ist. Der Sensor reagiert auf Temperaturveränderungen in seinem Sichtfeld. Somit können Menschen oder Tiere im Aktionsradius des Sensors erkannt werden. Jedoch kann der Sensor nur Veränderungen wahrnehmen. Bleibt ein Objekt ruhig im Bereich des Sensors stehen, so wird es nicht weiter erkannt. Sobald es sich weiterbewegt, schlägt der Sensor erneut an.[4, S. 493]

Im inneren eines solchen Moduls befinden sich zwei Folien, die an ihrer Oberfläche unterschiedliche elektrische Ladungen aufweisen. Trifft nun die Wärmestrahlung eines bestimmten Frequenzbereichs auf diese Folien, wird deren Polarisation verschoben und eine elektronische Spannung erzeugt, welche den Sensor zum auslösen bringt. Die milchige Kuppel auf dem Sensor erweitert den Erfassungsbereich des Sensors, indem es wie eine Anordnung von Linsen fungiert und lenkt die Wärmestrahlung direkt auf eine der beiden Folien. Bewegt sich nun eine Wärmequelle durch den vom Sensor überwachten Raum, kann man eine Bewegung über einen großen Bereich und in relativ großer Entfernung registrieren.[3] Um den PIR Sensor am Raspberry Pi anzuschließen, werden insgesamt 3 GPIO Anschlüsse benötigt. Einen 3,3V Anschluss für die Stromversorgung, einen Ground und ein frei programmierbarer GPIO Pin, um den PIR-Sensor zu steuern.

III. Codis: Kooperative, verteilte Überwachung

Um eine kooperative, verteilte Überwachung zu realisieren, haben wir das Programm Codis entwickelt. Codis steht für cooperative, distributed surveillance[1]. Entdeckt Codis einen Eindringling, sendet es eine INTRUDER Nachricht an alle Geräte im Netzwerk. Wie diese Nachricht verarbeitet wird und welche Maßnahmen eingeleitet werden sollen, wenn ein Eindringling entdeckt wurde, ist den Anwendern von Codis überlassen. In diesem Abschnitt befassen wir uns damit, wie Codis als verteiltes System ein Gebiet auf Eindringlinge überwacht. Dazu klären wir, wie Codis einen Eindringling entdeckt und wie aus Codis ein verteiltes System entsteht. Anschließend stellen wir fest, wie Codis als verteiltes System seine Effizienz verbessert und seine Genauigkeit beim Entdecken von Eindringlingen erhöht.

A. Bewegungserkennung mit der Kamera

Der Raspberry verwendet einen H264 Kodierer zur Videokompression. Um Videos zu kodieren, verwendet der Kodierer ein Verfahren, das als Motion Estimation[2][6] bezeichnet wird. Dabei wird das Bild in 16x16 Pixel große Quadrate eingeteilt. Diese Quadrate bezeichnet man als Makroblöcke. Beim Kodieren von Videos vergleicht der H264 Kodierer das aktuelle Bild mit einem Referenzbild. Dazu untersucht der Kodierer jeden einzelnen Makroblock im aktuellen Bild und sucht nach dem ähnlichsten Makroblock im Referenzbild. Der jeweilige Abstand zwischen den Makroblock im aktuellen Bild und im Referenzbild wird vom Kodierer gespeichert. Anhand dieses Abstands kann gemessen werden, wie sehr sich ein Makroblock im Bild gegenüber dem Referenzbild bewegt hat. Bis zu diesem Schritt wurde der komplette Vorgang im H264 Kodierer implementiert, um Bewegungen zu erkennen.[8] Als nächstes nehmen wir uns für jedes Bild die Bewegungsdaten als Array und berechnen damit das Ausmaß an Bewegung, die im Bild stattfindet. Dazu berechnen wir das Ausmaß der einzelnen Vektoren im Array mit dem Satz des Pythagoras'. Eine Bewegung wurde dann erkannt, wenn im Array mindestens zehn Vektoren vorhanden sind, deren Bewegungsausmaß mindestens 60 beträgt.

B. Codis als verteiltes System

Codis' Hauptfunktion ist es, ein Gebiet aus verschiedenen Blickwinkeln zu überwachen. Dieser Ansatz soll die Genauigkeit beim Erkennen von möglichen Eindringlingen erhöhen. Dazu verwendet Codis mehrere Raspberrys, die sich in einem Netzwerk befinden und untereinander Nachrichten austauschen. Wir verwenden im Folgenden den Begriff Codis-System für die Menge aller Raspberrys, auf denen Codis läuft und die miteinander in einem Netzwerk kommunizieren. Als Koordinator bezeichnen wir ein Raspberry im Codis-System,

[1] zu Deutsch: Kooperative, verteilte Überwachung
[2] wortwörtlich: Bewegungsvorhersage

der spezielle Aufgaben übernimmt, die wir im Laufe dieses Abschnittes klären. Der Koordinator ist allen Raspberrys im Codis-System bekannt und wird innerhalb des Codis-Systems ausgewählt. Das Codis-System wird dann erzeugt, wenn ein Raspberry Codis ausführt, während kein anderer Raspberry im Netzwerk Codis ausführt und wird dann zerstört, wenn Codis vom letzten Raspberry im Codis-System beendet wird.

1) Codis-Liste: Codis verwendet eine verteilte Liste der Netzwerkadressen aller Raspberrys im Codis-System. Als verteilte Liste bezeichnen wir eine Liste, die auf allen Geräten eines verteilten Systems lokal abgespeichert ist und stets redundant zu den lokal abgespeicherten Listen der jeweils anderen Geräten ist. Die verteilte Liste, die Codis verwendet, bezeichnen wir im Folgenden als Codis-Liste. Die Codis-Liste wird zu Koordinationszwecken zwischen den Raspberrys im Codis-System benötigt. Die Codis-Liste wird dann gebildet, wenn das Codis-System erzeugt wird. Möchte ein Raspberry dem Codis-System beitreten, sendet dieser eine JOINREQUEST Nachricht an das Codis-System. Daraufhin horcht der Raspberry fünf Sekunden lang auf eine JOINRESPONSE Nachricht, die von allen Raspberrys im Codis-System versendet wird. Die JOINRESPONSE Nachricht enthält die Position des Absenders in der Codis-Liste. Nachdem der Koordinator seine JOINRESPONSE Nachricht versendet hat, wartet er eine Sekunde und sendet daraufhin eine COORDINATOR Nachricht an den Raspberry. Die COORDINATOR Nachricht, enthält die Position des Koordinators in der Codis-Liste. Erhält der Raspberry die COORDINATOR Nachricht, dann trägt er ein, welche Position der Koordinator in der Codis-Liste hat. Erhält der Raspberry nach fünf Sekunden keine JOINRESPONSE Nachricht, trägt er sich als Erster in die Codis-Liste ein und ist somit auch der Koordinator im Codis-System. Hat der Raspberry von allen anderen Raspberrys im Codis-System eine JOINRESPONSE Nachricht erhalten, trägt er sich ans Ende der Codis-Liste ein und sendet eine JOIN Nachricht an alle Geräte im Codis-System. Erhält ein Raspberry eine JOIN Nachricht, trägt er den Absender der JOIN Nachricht ans Ende seiner Codis-Liste ein.

2) Wahl eines Koordinators: Der Raspberry von dem aus das Codis-System erzeugt wurde, wird als erster Koordinator ausgewählt. Betreten weitere Raspberrys das Codis-System, wird ein modifizierter Ringalgorithmus[7][2] ausgeführt, der alle 15 Minuten einen neuen Koordinator auswählt. Um einen logischen Ring darzustellen, verwendet Codis die Codis-Liste. Die Raspberrys im Codis-System sind anhand ihrer Position in der Codis-Liste aufsteigend, im Ring angeordnet. Wird ein neuer Koordinator ausgewählt, verschickt der derzeitige Koordinator eine ELECTION Nachricht, an seinem Nachfolger im Ring. Erhält ein Raspberry eine ELECTION Nachricht, dann trägt er sich als neuer Koordinator ein und sendet eine COORDINATOR Nachricht an alle Raspberrys im Codis-System. Empfängt der Raspberry, der die ELECTION Nachricht gesendet hat, nach fünf Sekunden keine COORDINATOR Nachricht, sendet er eine HEARTBEATREQUEST an seinen Nachfolger. Erhält er daraufhin eine HEARTBEATRESPONSE Nachricht zurück, sendet er die ELECTION Nachricht erneut an seinen Nachfolger. Empfängt er dagegen keine HEARTBEATRESPONSE von seinem Nachfolger, dann entfernt er diesen aus der Codis-Liste und sendet eine LISTUPDATE Nachricht, an das Codis-System, damit der Nachfolger aus der Codis-Liste entfernt wird. Danach versucht der Raspberry die ELECTION Nachricht, an seinen nächsten Nachfolger zu senden.

3) Abwechselnde Überwachung: Unter abwechselnder Überwachung verstehen wir, dass nur der Koordinator das Gebiet überwacht, bis der nächste Koordinator gewählt wurde. Währenddessen gelten alle anderen Raspberrys als inaktiv. Inaktive Raspberrys haben ihre Sensoren abgeschaltet und warten auf eine ELECTION Nachricht. Entdeckt der Koordinator einen möglichen Eindringling, dann sendet dieser eine INTRUDER Nachricht an alle anderen Raspberrys und geht in einen Alarmzustand über. Empfängt ein inaktiver Raspberry diese Nachricht, geht dieser auch in einen Alarmzustand über. Im Alarmzustand sind die Sensoren des Raspberrys aktiviert. Entdeckt ein Raspberry im Alarmzustand für fünf Minuten keinen Eindringling, dann geht er wieder in den inaktiven Zustand, außer der Koordinator der den Alarmzustand lediglich verlässt. Wird ein Raspberry im Alarmzustand zum Koordinator, verbleibt er im Alarmzustand. Wird ein neuer Koordinator gewählt, verbleibt der vorherige Koordinator im Alarmzustand, falls er sich in diesem bereits befindet.

C. Entdecken eines Eindringlings

Entdeckt Codis in mehreren Bildern für einen kurzen Abstand eine Bewegung, setzt es einen speziellen Wert auf TRUE. Dieser spezielle Wert wird nach 50 Bildern, in denen keine Bewegung erkannt wurde, auf FALSE gesetzt. Ein möglicher Eindringling wird dann erkannt, wenn der PIR-Sensor Bewegungen dann erkennt, während dieser spezielle Wert auf WAHR gesetzt ist. Entdeckt ein Raspberry im Codis-System einen möglichen Eindringling, dann sendet dieser eine INTRUDER Nachricht an den Koordinator. Empfängt der Koordinator eine INTRUDER Nachricht, merkt er sich, von welchem Raspberry diese Nachricht versendet wurde und wann die Nachricht versendet wurde. Entdeckt der Koordinator einen möglichen Eindringling, dann prüft er, ob ein anderer Raspberry in den letzten drei Sekunden einen möglichen Eindringling entdeckt hat. Ist das der Fall, sendet der Koordinator eine INTRUDER Nachricht ans Netzwerk.

IV. Andere Ansätze (bzw. Verwandte Arbeiten)

Zu Beginn des Projekts haben wir uns stark mit dem Programm Motion auseinandergesetzt. Mit Motion ist es möglich, seinen (Linux-)Rechner mit USB-Kamera in eine Überwachungskamera mit vielen zusätzlichen Funktionen zu verwandeln und war aus diesem Grund für uns von Interesse. Die Konfigurationsmöglichkeiten sind sehr umfangreich und die Bewegungserkennung in der Software hat auf Wunsch Videos und/oder Fotos aufgezeichnet, sobald der Algorithmus eine Bewegung wahrgenommen hat. Bei der Aufzeichnung von Videos können die Auflösung und die Framerate selber bestimmt werden, bei Fotos kann ebenfalls die Auflösung an eigene Bedürfnisse angepasst werden. Auf Wunsch besteht auch die Möglichkeit, sich einen http-Stream zur Verfügung stellen zu lassen, mit welchem die Möglichkeit

besteht, das aktuelle Geschehen vor der Kamera auch aus der Entfernung zu beobachten. Das Programm verzichtet vollständig auf eine grafische Oberfläche und kann als Daemon im Hintergrund ausgeführt werden. Also muss man auch als lokaler Anwender auf den http-Stream zugreifen, um das aktuelle Bild abzugreifen und das Geschehen dort einzusehen. Da Motion mit handelsüblichen USB-Webcams in Betrieb genommen werden kann, konnten wir das Programm sofort erproben:
Wir haben die Softwarepakete zu Motion also auf den Raspberry Pi installiert, eine USB-Webcam angeschlossen und konnten die Bewegungserkennung testen. Um eine ausreichende Qualität der bei Bewegung automatisch erstellten Fotos und Videos zu erreichen, musste die Auflösung hoch genug sein, was zu ersten Problemen führte: Der Raspberry Pi war mit der Speicherung der Daten sehr ausgelastet und auch der Algorithmus zur der Bewegungserkennung forderte bei entsprechend hoher Auflösung, und damit einhergehender Genauigkeit, seinen Tribut. Das System wäre dauerhaft bei einer CPU-Auslastung zwischen 80 - 100%. Zu diesem Zeitpunkt betrachteten wir dies als notwendiges übel und fuhren trotzdem fort. Um den Anforderungen unseres Projekts gerecht zu werden, auch bei Nacht Bewegungen zu erkennen, musste allerdings ein anderes Kamerasystem genutzt werden. Das bereits vorgestellte Raspberry Pi Kameramodul wurde diesen Anforderungen gerecht, allerdings unterstützt Motion, wie erwähnt, nur USB-Kameras. In der Raspberry Pi Community war man sich dessen bereits bewusst, weshalb eine modifizierte Version von Motion existiert, mit welcher auch das Raspberry Pi Kamera Modul genutzt werden kann. Auch hier gab es leider Probleme mit der Performance, ein anderes Problem führte allerdings erst dazu, dass wir die Arbeiten mit Motion letztlich eingestellt haben: Sowohl vor, als auch nach der Modifizierung des Quellcodes von Motion gab es bei der Kompilierung viele Systemseitige Fehler, welche auch nach viel investierter Zeit nicht vollständig auszumerzen waren. Die Konfigurationsdatei zu verändern, mit welcher z.B. die Auflösung und der http-Stream eingestellt werden konnten, half uns bei dem Ziel nicht weiter, ein verteiltes Überwachungssystem zu erstellen und wir waren nicht in der Lage, den Pir-Sensor mit Motion überhaupt anzusteuern. Das Ziel war nicht erreichbar ohne den in C geschriebenen Quellcode auch dahingehend anzupassen. Um einem erneuten Rückschlag vorzubeugen haben wir uns dann dazu entschlossen, mit der Programmierung eines eigenem Python-Codes zu beginnen. Die aus dem Motion-Quellcode bezogenen Erfahrungen zu dem Algorithmus der Bewegungserkennung konnten wir für unseren eigenen Ansatz gut berücksichtigen. Im Internet konnte ein recht minimalistischer Beispielcode für die Bewegungserkennung mit dem Raspberry Pi Kamera Modul gefunden werden[1] und durch die öffentlich zugänglichen Projekte aus der Raspberry-Community konnten Lehren daraus gezogen werden, wie man einen möglichst effektiven Algorithmus schreibt, der unseren Ansprüchen genügt. In dem Buch R̈aspberry Pi - Das umfassende HandbuchK̈onnte ein Beispiel für das Arbeiten mit dem PIR-Sensor gefunden werden[**Raspberry**], welchen wir teilweise in unseren Code übernehmen konnten. Die Ideen und Beispiele haben wir dazu genutzt, überhaupt ein Gespür für die Programmierung mit Python zu erlangen und unsere eigenen Ideen endlich in das Projekt einbringen zu können.

V. Evaluation

Codis soll als verteiltes System nicht nur seine Genauigkeit beim Entdecken von Eindringlingen erhöhen, sondern auch die CPU-Auslastung der einzelnen Raspberrys verbessern. Dazu ist Codis' Funktion der abwechselnden Überwachung gedacht. Überwacht ein Raspberry mit Codis das Gebiet, beansprucht er ungefähr 30 Prozent an CPU Leistung, ohne einen Eindringling zu entdecken. Durch die abwechselnde Überwachung, teilen sich die Raspberrys die CPU-Last auf, indem nur einer für ein Intervall von 15 Minuten das Gebiet überwacht, bis ein Eindringling entdeckt wurde. Die Raspberrys, die mit Codis ein Gebiet im Freien überwachen, müssten mit externen Akkus betrieben werden. Durch die abwechselnde Überwachung wird der Akku eines Raspberrys weniger beansprucht.

VI. Zusammenfassung

Für jeden von uns war das Thema Raspberry Pi ein nahezu unbeschriebenes Blatt. Von der Existenz des Gerätes haben wir gewusst, welche Ideen mit dem Gerät verwirklicht werden können, beginnen wir gerade zu begreifen. Nachdem wir anfangs davon ausgingen, mit Motion bereits ein sehr gut funktionierendes Softwaregerüst entdeckt zu haben, haben wir dementsprechend hohe Ziele gesteckt. Immerhin konnte davon ausgegangen werden, dass es nicht nötig war sich um den Algorithmus zur Bewegungserkennung und anderen Funktionen selbst zu kümmern. An den richtigen Stellen im Quellcode eingesetzt, hätten viele kleine Features (wie ein Motorbetriebener Schwenkvorgang der Kamera) verwirklicht werden können, denn für viele Features ist Motion auch bereits gut genug designed, sodass die Implementierung leicht gewesen wäre. Aber auch der Weg hin zu der Erkenntnis, im Endeffekt selbst den Code schreiben zu müssen, war sehr lehrreich. So tiefgehend hat sich keiner aus unserer Gruppe jemals mit einem Linux-System auseinander gesetzt. Die Installation diverser Programme, wie dem VNC-Server oder anderen, zugegebenermaßen für private Zwecke genutzten Dingen wie dem Music Player Daemon, haben ein größeres Verständnis für Linux hervorgerufen und auch aufgezeigt, was ein Linux-System im Gegensatz zu einem Windows-System aus macht. Viele nützliche Sachen kommen ohne Grafische Oberfläche aus und lassen sich nur über Text-Dateien konfigurieren, sind dafür aber bis ins kleinste Detail konfigurierbar. Wenn für die Installation eines Paketes bestimmte andere Pakete vorhanden sein müssen, installiert das System die nötigen gleich mit. Diese positiven Eigenheiten sind natürlich nur exemplarisch herausgezogen.
Nicht nur das Betriebssystem hat sich mir/uns durch das Projekt weiter geöffnet. Auch der Rapsberry Pi hat sich durch seine kompakte Bauform für viele weitere Projekte interessant gemacht. Die Community um den Raspberry Pi hat sich schon vieler nützlicher Projekte angenommen. So ist es zum Beispiel

ein Leichtes, den Raspberry Pi in eine Retro-Spielekonsole oder einen Medienserver zu verwandeln (für letzteren Verwendungszweck gibt es sogar bereits ein eigenes Betriebssystem). Durch das GPIO-Board ist es aber auch für Hobby-Robotik ein perfektes System (Stichwort: Automatischer Geschützturm (mit Schaumstoffraketen natürlich)).
Aber zurück zu Codis: Durch den Rückschlag mit Motion haben wir uns mit Python auseinander setzen müssen, viele ursprünglich geplante Features fallen gelassen aber blicken dafür auf eine völlig Eigenständige Lösung für die Aufgabenstellung zurück: Der Code ist lediglich durch Versatzstücke anderer Projekte inspiriert und zeichnet sich durch eine einzigartige Lösung für das verteilte, kooperative Überwachen eines Gebiets auf Eindringlinge aus.

LITERATUR

[1] Okt. 2014. URL: https : / / github . com / waveform80 / picamera/blob/master/docs/recipes2.rst.

[2] George Coulouris, Jean Dollimore und Tim Kindberg. *Verteilte Systeme : Konzepte und Design*. München : Pearson Studium, 2003.

[3] *Elektronikwissen zu: PIR-Sensor - Aufbau und Funktion*. URL: http://www.elv.de/controller.aspx?cid=758&detail=10&detail2=129.

[4] Michael Kofler. *Raspberry Pi : das umfassende Handbuch ; [Grundlagen verstehen, spannende Projekte realisieren ; Schnittstellen des Pi, Schaltungsaufbau, Steuerung mit Python ; Erweiterungen für den Pi: Gertboard, PiFace, Quick2Wire u.a. in Hardware-Projekte einsetzen ; aktuell zu allen Versionen, inkl. Modell B+]*. Bonn : Galileo Press, 2014.

[5] Mirko Lindner. *Über 3,8 Millionen Raspberry Pi verkauft*. Okt. 2014. URL: http://www.pro-linux.de/news/1/21613/ueber-38-millionen-raspberry-pi-verkauft.html.

[6] Shilpa Metkar und Sanjay Talbar. *Motion Estimation Techniques for Digital Video Coding*. Springer India, 2013.

[7] Andrew S. Tanenbaum und Martinus R. van Steen. *Verteilte Systeme : Grundlagen und Paradigmen. Distributed Systems ¡dt.¿* München : Pearson Studium, 2003.

[8] Upton, Liz. *Vectors from coarse motion estimation*. URL: http://www.raspberrypi.org/vectors-from-coarse-motion-estimation.