Discoverpixy

Ein Projekt zur Objekterkennung mittels der
Pixy CMUCam5 und der STM32 Mikrokontroller-Plattform

Autoren:	Aaron Schmocker und Timo Lang
Dozent:	G. Krucker
Modul:	Softwaredesign und Softwareprojekte (BTE5052)
Abgabedatum:	1.6.2015

Inhaltsverzeichnis

1 Aufgabenstellung

Im Rahmen des Moduls BTE5052 soll eine Projektarbeit durchgeführt werden. Ziel ist es mithilfe der Videokamera Pixy CMUCam5 Objekte zu erkennen, ihnen zu folgen, die Erkennung geeignet darzustellen und zu konfigurieren. Als Entwicklungsplattform soll die STM32 Mikrokontroller-Serie dienen. Die effektive Objekterkennung erfolgt durch die Pixy-Kamera und die darunterliegende Technologie sowie die Thematik der Bildverarbeitung sei nicht Ziel dieser Arbeit. Mit der Pixy-Kamera ist via USB oder I2C zu kommunizieren. Als Miktokontroller-Kit stehen das STM32F4Discovery Board und das Carme-Kit (BFH) zur Verfügung. Die erstellte Software ist in der Programmiersprache C zu schreiben und entsprechend zu kommentieren und dokumentieren.

Wir werden für die Realisierung dieser Arbeit das STM32F4Discovery Board verwenden und via USB mit der Pixy-Kamera kommunizieren. Um die Entwicklung zu vereinfachen wird nebenbei ein Emulator enwickelt, welcher das Ausführen der Applikation auf dem PC erlaubt.

2 Planung und Ziele

Fokus der Arbeit:

- Planung und Durchführung eines Mikrokontroller-Projekts im Zweierteam.
- Design einer erweiterbaren Architektur, so dass das Projekt von Dritten weitergeführt und optimiert werden könnte.
- Implementierung und Dokumentation einer funktionstüchtigen Lösung gemäss Aufgabenstellung.
- Abstraktion der Anwendung, zur Ausführung auf dem PC bzw. dem STM32F4Discovery

Nicht Fokus der Arbeit:

- Korrekte und schöne Implementierung des Usb-Hosts gemäss Spezifikation für die Pixy-Kamera auf dem STM32F4Discovery.
- Herstellen von produktionsreifen Bibliotheken zur Kommunikation mit Display, SD-Karte und Pixy
- Herstellen eines komplexen Emulators, dessen Funktionalität über die Funktionalität der Anwendung auf dem STM32F4Discovery hinausgeht.
- Perfomance-Optimierungen

Zeitplanung:

Name	Bemerkungen	Datum
FSMC Display Funktionen	Mindestens diese die auch im Emulator vorhanden sind, Ziel: Pixy video auf dem Display	20.4.2015
Touch Controller, Basic	Ziel: zeichnen von Pixel auf dem Display via Touch	27.4.2015
Sd Charte, Basic	Ziel: anzeigen eines Bitmaps auf dem Display	27.4.2015
2nd Display support	Nur wenn auch via FSMC möglich.	27.4.2015
Speisung		27.4.2015
Schaltplan		27.4.2015
Hardware ok		4.5.2015
Display fertig abstrahiert		4.5.2015
Touch fertig abstrahiert		4.5.2015
Sd Karte fertig abstrahiert		4.5.2015
Gui Library fertig	d.h. Buttons, Chechbox, Slider und ev NumUpDown, Radio, Dropdown	11.5.2015
Alle LowLevel funktionalitäten fertig	Ziel: ab hier wird nur noch der “common”-Order verwendet	11.5.2015
Objekte von Pixy empfangen	Ziel: die von Pixy erfassten Objekte werden auf dem Display dargestellt	18.5.2015
Objektracking	Ziel: Objekttracking funktioniert und ist konfigurierbar	25.5.2015
Abgabe		1.6.2015

3 Analyse

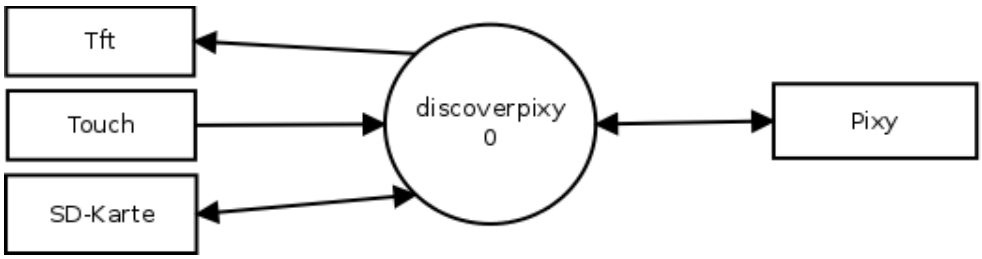

Illustration 1: Kontextdiagramm

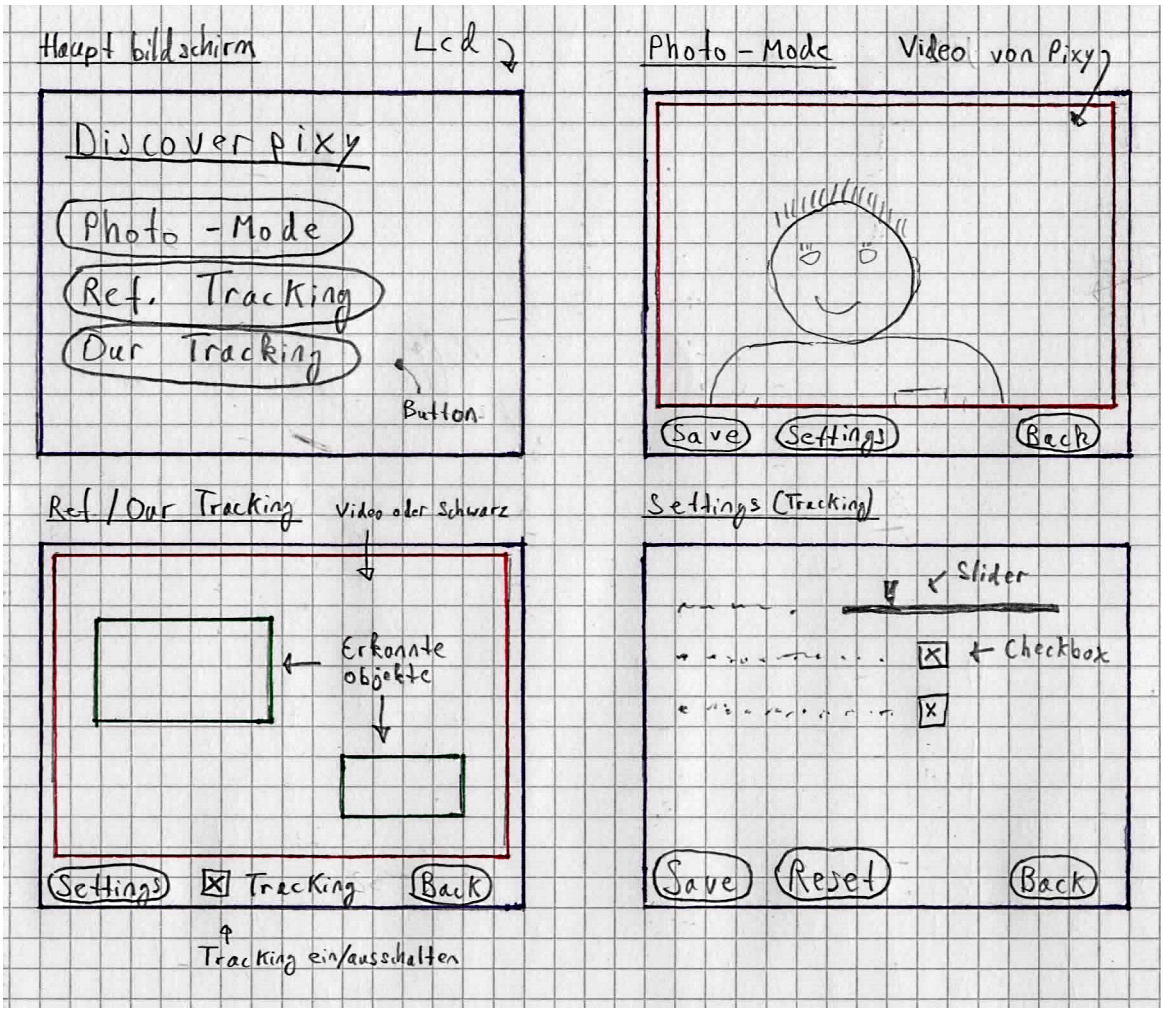

Illustration 2: Ein Prototyp verschiedener Menüs und Untermenüs

4 Grobdesign

Die Software soll in Module unterteilt werden. Das folgende Diagramm zeigt die vorhandenen Module und deren Abhängigkeiten. Findet eine Kommunikation zwischen 2 Modulen statt so wird dies durch einen Pfeil dargestellt.

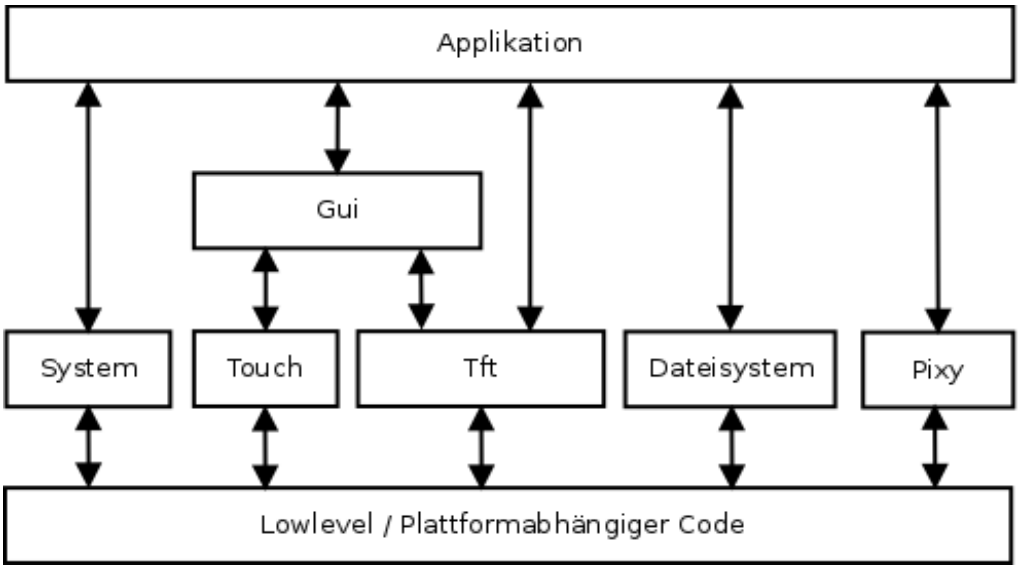

Illustration 3: Übersicht über die Module und deren Abhängigkeiten

Alle Module sollen aus Plattformunabhängigem Code bestehen. Die Module greifen wenn nötig auf Plattformspezifischen Code zu. Dieser wird aber abstrahiert und muss dann für jedes Zielsystem separat implementiert werden. Details zu der Implementierung folgen in einem späteren Kapitel. Im folgenden soll jedes Modul und seine Schnittstellen kurz beschrieben werden

4.1 Applikation

Ordername:	app
Beschreibung:	Das Applikations Modul beinhaltet den effektiven Code zur Steuerung aller Geräte und Interaktion mit dem Benutzer.
Benötigte Module:	System, Gui, Tft, Dateisystem, Pixy
Benötigt von:	Keinem Modul, aber dem Gesamtsystem.
Benötigte Low-Level Funktionen:	Keine
Bereitgestellte Funktionen:	void app_init(); void app_process();

4.2 System

Ordername:	system
Beschreibung:	Das System Modul kontrolliert die darunterliegende Hardware und bietet Sleep-Funktionen an
Benötigte Module:	keine
Benötigt von:	Applikation

Benötigte Low-Level Funktionen:	bool ll_system_init();
	void ll_system_delay(uint32_t msec);
	void ll_system_process();

Bereitgestellte Funktionen:	bool system_init();
	void system_delay(uint32_t msec);
	void system_process();

4.3 Tft

Ordername:	tft
Beschreibung:	Das Tft Modul ermöglicht das Zeichnen von Formen und Texten auf ein Display.
Benötigte Module:	keine
Benötigt von:	Gui, Applikation

Benötigte Low-Level Funktionen:	bool ll_tft_init();
	void ll_tft_clear(uint16_t color);
	void ll_tft_draw_line(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color);
	void ll_tft_draw_pixel(uint16_t x,uint16_t y,uint16_t color);
	void ll_tft_draw_rectangle(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2, uint16_t color);
	void ll_tft_fill_rectangle(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2, uint16_t color);
	void ll_tft_draw_bitmap_unscaled(uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t *dat);

Bereitgestellte Funktionen:	bool tft_init();
	void tft_clear(uint16_t color);
	void tft_draw_line(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color);
	void tft_draw_pixel(uint16_t x,uint16_t y,uint16_t color);
	void tft_draw_rectangle(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2, uint16_t color);
	void tft_fill_rectangle(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2, uint16_t color);
	void tft_draw_bitmap_unscaled(uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t* dat);

4.4 Touch

Ordername:	touch
Beschreibung:	Das Touch Modul empfängt Benutzereingaben und löst Benachrichtigungen im

	Falle einer Interaktion aus.
Benötigte Module:	keine
Benötigt von:	Gui
Benötigte Low-Level Funktionen:	bool ll_touch_init();

Bereitgestellte Funktionen:	bool touch_add_raw_event(int x, int y, Event_Flags flags);
	bool touch_init();
	bool touch_has_empty(unsigned char num);
	bool touch_register_area(TOUCH_AREA_STRUCT* area);
	bool touch_unregister_area(TOUCH_AREA_STRUCT* area);

4.5 Dateisystem

Ordername:	filesystem
Beschreibung:	Das Dateisystem Modul bietet Funktionen zum lesen und schreiben auf einen Datenträger (z.B. SD-Karte)
Benötigte Module:	keine
Benötigt von:	Applikation
Benötigte Low-Level Funktionen:	bool ll_filesystem_init(); …. noch unklar
Bereitgestellte Funktionen:	bool filesystem_init(); … noch unklar

4.6 Gui

Ordername:	gui
Beschreibung:	Das GUI Modul bietet Funktionen zum erstellen von und interagieren mit, Gui-Elementen (z.B. Buttons).
Benötigte Module:	Tft, Touch
Benötigt von:	Applikation
Benötigte Low-Level Funktionen:	keine

Bereitgestellte Funktionen:	bool gui_init();
	bool guiAddButton(BUTTON_STRUCT* button);
	void guiRemoveButton(BUTTON_STRUCT* button);
	void guiRedrawButton(BUTTON_STRUCT* button);
	bool guiAddBitmapButton(BITMAPBUTTON_STRUCT* button);
	void guiRemoveBitmapButton(BITMAPBUTTON_STRUCT* button);

	void void guiRedrawBitmapButton(BITMAPBUTTON_STRUCT* button); .. und noch mehr

4.7 Pixy

Ordername:	pixy
Beschreibung:	Das Pixy Modul bietet Funktionen zur Interaktion mit der Pixy Cam.
Benötigte Module:	keine
Benötigt von:	Applikation
Benötigte Low-Level Funktionen:	Keine ll_* funktionen. Alle bereitgestellten Funktionen werden direkt von der Plattform-Implementierung bereitgestellt.

Bereitgestellte Funktionen:	int pixy_init();
	int pixy_blocks_are_new();
	int pixy_command(const char *name, ...);
	void pixy_close();
	int pixy_led_set_RGB(uint8_t red, uint8_t green, uint8_t blue);
	int pixy_led_set_max_current(uint32_t current);
	int pixy_led_get_max_current();
	int pixy_cam_set_auto_white_balance(uint8_t value);
	int pixy_cam_get_auto_white_balance();
	uint32_t pixy_cam_get_white_balance_value();
	int pixy_cam_set_white_balance_value(uint8_t red, uint8_t green, uint8_t blue);
	int pixy_cam_set_auto_exposure_compensation(uint8_t enable);
	int pixy_cam_get_auto_exposure_compensation();
	int pixy_cam_set_exposure_compensation(uint8_t gain, uint16_t compensation);
	int pixy_cam_get_exposure_compensation(uint8_t * gain, uint16_t * compensation);
	int pixy_cam_set_brightness(uint8_t brightness);
	int pixy_cam_get_brightness();
	int pixy_rcs_get_position(uint8_t channel);
	int pixy_rcs_set_position(uint8_t channel, uint16_t position);
	int pixy_rcs_set_frequency(uint16_t frequency);
	int pixy_get_firmware_version(uint16_t * major, uint16_t * minor, uint16_t * build);

5 Implementierung der Module

Es müssen vorallem die Teile beschrieben werden die nicht direkt an LowLevel weitergeleitet werden

5.1 Applikation

Aufbau der App erklären, die einzelnen Screens & Funktionalitäten erklären

Unser Pixy PID regler erklären

State machine?

5.2 System

Trivial, wird alles weitergeleitet

5.3 Tft

Vermutlich Trivial, wird bisher alles weitergeleitet

5.4 Touch

Callbacks erklären. Vererbung in C erklären. Touch library vorstellen. Quellen angeben

5.5 Dateisystem

Vermutlich Trivial, wird alles weitergeleitet

5.6 Gui

Gui library vorstellen. Quellen angeben (timo bms)

5.7 Pixy

Mehr oder weniger trival. Implementierung ist komplett plattformabh.

Erklären was an der Pixy original software für BEIDE implementierungen verändert wurde (multithreading entfernt, etc)

6 Plattform STM32F4Discovery

6.1 System

Stm beschreiben. Initialisierungs prozedere (sysclock, systick)

6.2 Tft

Lcd ansteuerung beschreiben. Auf controller verweisen. Quellen angeben (timo bms, fsmc aus dem netz)

6.3 Touch

SPI erklären, Controller ansteuerung beschreiben. Quellen angeben (timo bms)

6.4 Sd-Karte

SPI erklären, Controller ansteuerung beschreiben. Quellen angeben (fatfs)

6.5 Usb & Pixy

STM USB erklären, Pixy “Link”-Implementierung beschreiben. Quellen angeben (Stm usb lib, Pixy)

7 Plattform Emulator

7.1 System

Emulator & qt beschreiben

7.2 Tft & Touch

Mainwindow und rendering beschreiben

7.3 Dateisystem

7.4 Usb & Pixy

Auf libusb verweisen und Pixy “Link”-Implementierung beschreiben