The Capsorter

Developed by Jacob Biscan and Jonah Meggs at UNSW (August-December 2019), continued by Jonah Meggs at GreenBatch (December 2019 – February 2020).

Contents

[Preface 2](#_Toc33174480)

[Hopper 2](#_Toc33174481)

[Conveyor belt 2](#_Toc33174482)

[Singulation 3](#_Toc33174483)

[Size rejection 3](#_Toc33174484)

[Guide rails 3](#_Toc33174485)

[Image Analysis 3](#_Toc33174486)

[ml.exe 3](#_Toc33174487)

[capsorter.exe 4](#_Toc33174488)

[Ejection 4](#_Toc33174489)

[main.c 4](#_Toc33174490)

[Solenoids 5](#_Toc33174491)

[Power Supply 6](#_Toc33174492)

[Software Installation 6](#_Toc33174493)

[Compilation 7](#_Toc33174494)

[Capsorter 7](#_Toc33174495)

[ml 7](#_Toc33174496)

[main 7](#_Toc33174497)

[Loading code to ATMEGA8U2 7](#_Toc33174498)

[Troubleshooting 8](#_Toc33174499)

[GUI manual 8](#_Toc33174500)

[Conclusion 8](#_Toc33174501)

# Preface

Caps adhere to Murphy’s Law in that if they can jam they will jam.

# Hopper

The hopper is a box that gravity feeds caps into an auger. The dimensions are non-specific but it is important to have the hopper narrower at the bottom so that the caps naturally move towards the auger (instead of straight down). This has been achieved by a board and tape in the current model.

The auger is 20mm (I think) PVC with Perspex flights. The pitch of the auger is about 100mm, diameter is 200mm. A 150mm diameter auger was also trialled (with a smaller pitch), however caps such as Cadbury purple caps and laundry caps jammed in the auger. The 200mm auger has so far had no issues (apart from it breaking off the axle due to a horrible glue job). The auger needs to be angled up slightly so that each pitch is not full with caps, and so that they get gradually pushed off the lip of the shaft (if angled downwards, flights become full and too many caps are pushed onto the singulator at once).

The control box, motor, chain and gears were stolen from a small 2m conveyor belt, probably need to be returned at some point. The control box also powers 2 other larger conveyor belts Greenbatch owns, so if it goes missing ask Natasha or Damo. The gear is attached to the axle by a nut (diagonally scored with a grinder) that was inserted into the end of the axle when it was heated, and glued in place with Araldite. Bolt should come out to allow removal of gear. The motor is generally set to turn as slow as possible. To remove chain, remove one of the nuts securing the motor to the stand, and pivot motor to give chain slack.

Materials: Perspex flights were cut with scroll saw, heated with hot air gun, and carefully bent. 2 flights were then screwed together so that the auger has 2 rotations. uPVC glue was used to bond the Perspex to the axle. Note: flights have come loose from axle, I have smothered them in glue but there is the potential they will come loose again, I will send a 200mm metal auger I have at home to GreenBatch when I get a chance.

# Conveyor belt

The system works with 2 different speed conveyor belts running parallel. This is achieved by having an axle with two different radii. The smaller radii drives the slower belt, whereas the larger radii drives the faster belt. The current radii are 25mm and 90mm (PVC pipe), giving a speed increase of roughly 3. All inner axles are 25mm PVC, with skateboard bearings inserted to enable them to roll. The skateboard bearings are a tad small, thin strips of electrical tape were wound around the bearings to make them a snug fit. The rollers can be removed by taking off the nuts at either end (don’t let the bolts fall into the axle) and unscrewing one of the brackets. Alignment of the belts is suboptimal, hence guiding washers have been used in various places, in others the belt relies on guiding rails either side. The main axle is driven with a fan motor on the lowest setting stepped down a lot with bike sprockets and bike chain. This will hopefully be replaced with a similar motor to the one on the hopper at some point. The conveyor belt is signage material, there is heaps more of it if needed. It is important that the tape used on the fast conveyor belt is white so that the image processing is not affected. The frame is chipboard, edges under the conveyor belt have been filed to a curve, and is supported by school desk frames. The desk material is in the workshop, if the desks could be reassembled at some point that would be appreciated 😊.

For more durable conveyor belt material, Gavin at 0418938351 does small scale belts, ballpark $30 per metre. He is down in Rockingham.

# Singulation

## Size rejection

The current image processing technique relies on caps being flat on the conveyor belt. This means that caps such as pop tops, Gatorade caps, laundry caps etc. must not make it through to the camera. To achieve this there is a rotating blade that knocks large or irregularly shaped caps off the belt (and through a hole which is currently too small). The blade is at an angle in order to move the caps sideways off the belt. A simple bar was trialled however caps jammed under it. The motor used is taken from a printer, there is at least one spare. It runs on DC 5-8V depending on desired speed.

## Guide rails

Singulation is achieved by 2 guide rails that channel the caps first onto the slow conveyor, then onto the fast conveyor. This second transition is what pulls the caps into a single line, mostly with a gap between sequential caps. An arc has been attached to the second rail in an attempt to ensure the caps end up exactly lined up with the solenoids, however this only works to an extent, it is suggested that this arc be extended when there is more room between the rail and image processing unit. There is also a small ramp under the slow conveyor just before the second guide rail, this is to help caps move onto the faster belt (otherwise caps don’t move across, or worse get stuck underneath the second belt. A horrible issue is some of the juice caps have a lip on the bottom that loves to get under the conveyor belt it should be transitioning to, and then just sits there rotating, blocking other caps). This may not be an issue for a better quality conveyor belt.

Some jamming occurs at the tail of the first rail, it usually clears after 10 seconds or so. Cream lids and keg caps jam at the tail of the second rail because the gap isn’t large enough for them to fit through. This is an unresolved issue at this point (I’ve run out of space on the belt to try fix this issue. Potentially if the 2nd guide rail is raised 5mm or so, cream caps and rings should fit underneath on the slow belt, and therefore not interfere). Cadbury caps generally tip off the side at the tail of the second rail.

# Image Analysis

There is a $5 Jaycar webcam (actually a business card scanner) sitting inside the box, it is roughly flush with the surface. There is an LED on the camera but it is tiny, so an LED strip is blue-tacked above the camera (running on 12V). There is a black line on the far wall, this is used by ml.exe

## ml.exe

Accesses the webcam and continuously grabs frames from it. It first calibrates by scanning the image for the black line aforementioned (12V led strip must be on). Upon finding it, it monitors the middle of the image at points in the middle of and just below the black line. Because these 2 points are black and white respectively, the intensity difference is quite large. The program monitors the intensity difference, and once it drops below a threshold, the program knows a cap is present in the middle of the frame. The program works left and right from these two points attempting to find the black line again, when it does the program knows it has found the edges of the cap. From this an offset is calculated (centre of the cap from centre of the image), and an average RGB value is calculated from a horizontal line at the height of the lower analysis point. This RGB, offset and the time the frame was taken are then sent to capsorter.exe

## capsorter.exe

This is the GUI. It features buttons that add/remove bins (a bin is a destination for a particular colour cap), undo, skip the current cap and close the program. The program receives info from ml.exe, and compares it to data it knows already. This is for each bin an average RGB value, the number of times that average has been calculated on, and individual R, G, B average values. It makes its decision based on the bin that has the smallest largest difference between a R, G or B value (take a moment, not a spelling error). Look at the code for clarification. Once this has been decided, the total time delay since first seeing the image is calculated, the offset adjusted and a message is sent over USB to the mcu containing the offset in mm and the bin number. The USB transaction takes around 1-2 ms consistently from my ASUS and so is negligible. However you can use Wireshark with USBPcap to verify this (contact me if you need instruction).

The average bin RGB values can be modified by clicking the bin when that colour cap is identified in the GUI.

One noticeable bug is that it will delete or screw up the save file (data.txt) if you close the program directly after updating/adding/removing a bin. To avoid this, always run another cap through afterwards, and without updating any bins close the program. I will work on fixing this. However if a save file is particularly important please make a copy.

# Ejection

## main.c

The microprocessor used is an ATMEGA8U2 SMD. This was chosen because of its availability, support and cheap price. It is soldered onto a breakout board, there are more chips and breakout boards (and header pins) should you need to make another one. This chip has built in USB hardware so minimal CPU time is spent processing USB requests (VUSB on a regular ATMEGA8 was trialled however the timer interrupts for the solenoid delays mucked with the USB transfers). The microcontroller will respond appropriately to the USB message, if not, capsorter.exe will print an error to the terminal. I have found that occasionally a particularly demanding solenoid will cause a reset, and for some reason this results in the mcu not responding to USB messages until both the mcu and capsorter.exe are reset. If this becomes a recurring theme, consider a larger decoupling capacitor or using the 3.3V supply for the mcu instead of stepping down the 24V to 5V (so isolate the analog and digital supplies).

The USB message contains the length travelled thus far by the cap as well as the appropriate solenoid. This length is subtracted from the distance to the solenoid, and this length is then converted to ms. A general delay and then a solenoid specific delay is applied, and then an item containing the solenoid number and the time it should be actuated is added to a priority queue.

There is a timer interrupt occurring every 1ms, the priority queue is checked to determine if a solenoid needs to be actuated at this time, if so it is actuated, and another item is added to the queue containing the negative of the solenoid number and a solenoid on duration delay.

There is a counter used to determine the “time”, incremented each interrupt entry, once this passes a set overflow value, the counter is reset to 0 and all priority values in the priority queue are decreased by the overflow amount. This is to avoid the counter actually overflowing.

## Solenoids

The ATMEGA8U2 has 23 I/O lines, one is the reset pin and another is a clock pin from memory, as such these are not used. One is used for a status LED, leaving 20 lines to drive 20 solenoids. If more solenoids are required a larger ATMEGA family chip is recommended, as long as it has the USB hardware (identified with a U straight after the number of the atmega (ie ATMEGA16U2 etc)).

PCBs have been designed (in Eagle) and produced for the driving circuits, they are modularized to drive 4 solenoids each to aid in scalability of the capsorter. The received PCBs are missing some traces (my fault), as such you will notice blue wires joining the red solenoid wires in the green PCB terminal blocks. The design has been updated to fix this, and is ready for order (single side PCB x10 about $25 from JLCPCB). Due to components arriving late for the PCBs there are double all the components (as I got some at short notice from Jaycar/Altronics) apart from the MOSFETs. The MOSFETs are very expensive so if replacing the PCBs perhaps desolder and reuse these. There are 2mm screw holes in the PCB, but feel free to modify position/size in the .brd file.

The crimp pins and connectors that aren’t used were intended for the IDC cable for ease of connection to both PCB and breadboard, however they arrived late hence the solder and Altronics connectors.

When crimping, use the orange cable stripper to remove the insulation from the IDC cable, use a Stanley knife to separate each strand to a length of around 6cm, and then for each wire, crimp and bend crimp off strip (should snap off after a few bends). Solder wire in crimp pins (shouldn’t require much solder). Then insert into connector, being mindful of orientation.

Solenoids have been extracted from Brother printers (not inkjets). They are usually located on the opposite side to the power input, on the bottom right. The large solenoids don’t work unfortunately, but the smaller ones are perfect. Royna from Total Green Recycling has been collecting them for me, if you would like to continue to scavenge them let me know and I will get you in contact with her. Whilst I’m on the topic, solenoid 9 is a dud, it can be replaced with the solenoid sitting on top of it, just place the same ferrous rod with paddle and spring in it.

There are occasional issues with caps getting stuck between the solenoid paddles and conveyor belt, or between the solenoid paddles and chutes. This shouldn’t be a problem once the issue regarding the jam at the mouth of the image processing box and the issue of caps bouncing off the far wall of the image processing box and ending up in line with the solenoid paddles (not good) are resolved.

To add/remove solenoids, unscrew metal angle from plywood arms and twist it a bit to lay it down on the belt. You have to self-tap the screws but its only aluminium so you should be able to simply screw some electronics screws into the drilled holes (there is a yoghurt container of electronics screws somewhere around the capsorter…). You will notice there are 2 pairs of holes for the remaining solenoids, choose the ones that are increments of 10cm from the small, working solenoids. The paddles are perspex, 20x30mm, cut with scroll saw. To attach to solenoid arm, use a hacksaw to cut a small way down the middle of the top side, widen with a small file and push arm into the gap. Occasionally the Perspex is too thick, in which case file it down slightly. Superglue in place with excessive glue (no too excessive though, I tried to ensure that the arm notch was covered in glue, refer to current paddles for a better idea). These would probably be better 3D printed though.

# Power Supply

The conveyor belt and hopper have individual 240V plugs. The rest of the capsorter is powered by the powerboard from one of the printers (in the white eucalyptus something bottle). It provides 8V, 24V, and 3.3V. Currently I am using the 8V to power the height rejector motor, and 24V to power everything else, however it would be good to attempt to power the atmega separately on the 3.3V line to avoid spikes on the 24V line associated with the solenoids mucking with the supply voltage to the mcu (currently there is a 5V regulator (7805 I think, or potentially a 5V 3A regulator) stepping down the 24V).

It would be nice to put some capacitors either side of the regulators, I will put these in schematic with appropriate values and if you could trial them for me that would be great.

Additionally the regulators get quite hot, and this can lead to limited current output if run for long periods of time, so for the final design a computer fan would work well to help dissipate the heat.

# Software Installation

To enable the program to be run (Windows XP or 7)

1. Pull entire capsorter project from git
2. Navigate to libusb/libusb-win32-bin-1.2.6.0/bin
3. Plug capsorter USB into computer (make sure its not the camera USB connection)
4. Turn on power supply to mcu
5. Run inf-wizard
6. Select the USB device called Capsorter
7. Choose a place to save the driver
8. Install the driver. There will be a message saying the driver has no signature, install anyway

To enable the program to be run (Windows 8 or 10)

Windows 8 and 10 really doesn’t like to install unsigned drivers, you will need to restart your computer in a specific mode to enable installation of the USB driver.

1. Pull entire capsorter project from git
2. Hold shift key while clicking Restart
3. Navigate to Troubleshoot->Advanced Options->Startup Settings->Restart
4. After restart select option 7
5. Navigate to libusb/libusb-win32-bin-1.2.6.0/bin
6. Plug capsorter USB into computer (make sure its not the camera USB connection)
7. Turn on power supply to mcu
8. Run inf-wizard
9. Select the USB device called Capsorter
10. Choose a place to save the driver
11. Install the driver. There will be a message saying the driver has no signature, install anyway

To enable program to be modified:

1. Install OpenCV as per the instructions in “OpenCV Install Instructions.txt”
2. Install other programs as per “Compilation software instructions.txt”
3. Modify Makefile in project folder to reflect the correct path to the OpenCV libraries and headers on line 22

# Compilation

## Capsorter

In cmd line navigate to project folder, type and run

*make*

If an error is given, type and run

*gcc -I ./libusb/libusb-win32-bin-1.2.6.0/include -L ./libusb/libusb-win32-bin-1.2.6.0/lib/gcc -O capsorter.c -o capsorter.exe -lusb -lgdi32*

## ml

In cmd navigate to project folder, type and run

*make ml*

## main

In cmd navigate to project folder->mcu. Type and run

*make*

# Loading code to ATMEGA8U2

1. Navigate to project folder->mcu
2. Ensure power supply for capsorter is off
3. Remove breadboard connector closest to computer on LHS of breadboard (goes to 3rd solenoid PCB) without disrupting any wires
4. Connect ISP programmer (small blue PCB) to 10 pin ISP cable coming from breadboard
5. Connect ISP USB cable to computer (ISP programmer should light up red)
6. In cmd type and run

*avrdude -p m8u2 -P usb -c usbasp -U flash:w:main.hex*

1. Once load has finished it will hopefully display say x bytes of flash verified, if not unplug usb cable from computer and call me
2. Unplug ISP USB from computer
3. Unplug ISP programmer from ISP IDC cable
4. Reinsert 6 pin header from 3rd solenoid PCB

To run the capsorter

1. Ensure that camera and capsorter are both connected via USB to computer
2. Turn on power supply for electronics
3. In cmd, navigate to project folder, type and run “capsorter”
4. GUI will appear, wait for text below image to say “Camera Ready”
5. Turn on conveyor belt (by plugging in)
6. Turn on hopper (can be loaded first)

To shutdown the capsorter

1. Turn off hopper
2. Turn off conveyor belt (unplug motor)
3. Close capsorter.exe by clicking the X button
4. Turn off power supply to capsorter electronics

# Troubleshooting

* When capsorter.exe is run, I get a dialog box saying something “The code execution cannot proceed because <file>.dll was not found. Reinstalling the program may fix this problem”

Call me and tell me the name of the .dll file, I will find one and update the repository.

* My save file disappeared!

I am aware of this issue and am in the process of fixing it, in the meantime ensure that before the program is closed a cap is run through the system and you do not update or create a bin using that information. (Somehow acts as a buffer)

* In the GUI, “Calibrating camera” never changes to “Camera Ready”

This is due to the wrong camera being selected (typically the webcam on your computer instead of the webcam via USB). If you have the ability to modify code, attempt to change the CAMERA\_ID define, and recompile. Typically it is 0 if the computer does not have a built in webcam, or 1 if the computer does have a built in webcam.

* No solenoids move even though the image processing appears to be working!

Have a look at the terminal. If you see messages like “Could not find USB device while transmitting bin x”, check that the capsorter is connected via USB to the computer, and if so, restart the system. This tends to occur when a solenoid causes a large voltage drop, which affects the 5v regulator.

* The images are really dark or don’t look like the inside of the box!

Check the LED strip inside the camera box, it’s just bluetacked to the wall so it may have fallen off

# GUI manual

Buttons in order:

Undo: undo the last action, up to 10 actions. This will undo changes to bins, addition of bins or deletion of bins.

Add/remove: after selecting, click on an existing bin to delete it or click on the + icon to add a new bin with the current colour

Skip: Not implemented yet, will skip the current cap

Close: saves data and closes program

When a cap moves through the system, and is identified by the program, you have the opportunity if you want to either create a new bin for that colour, or modify an existing bin. You would want to modify an existing bin if it doesn’t correctly guess a colour. Modifying a bin takes the average RGB value, multiplies it by the number of times it has been modified before (n), adds the current cap RGB value and then divides by n+1. This is implemented by clicking on a bin once a cap has been recognised by the system.

# Conclusion

There are still many issues to fix with this prototype however I believe the basic capsorter functionality is present. I will continue to work on the GUI and schematics/PCB layouts, and commenting the code over the next few weeks. I wish you all the best in the continuation of this project, and would love to help you if you get stuck, want some inspiration, can’t understand my code or are trying to understand why I’ve put that random piece of metal there! I can be contacted on 0459709377, at [jmeggs@outlook.com.au](mailto:jmeggs@outlook.com.au), and would be very happy to video chat over messenger or similar to aid in troubleshooting. Thankyou and I look forward to hearing about your progress!

--Jonah Meggs Feb 2020